Unforgettable Fuzzy Extractor: Practical Construction and Security Model

Oleksandr Kurbatov[†], Dmytro Zakharov[‡], Lasha Antadze[†],
Victor Mashtalyar[‡], Roman Skovron[‡], Volodymyr Dubinin[‡]

[†]Rarimo [‡]Distributed Lab

October 1, 2025

Abstract

Secure storage of private keys is a challenge. Seed phrases were introduced in 2013 to allow wallet owners to remember a secret without storing it electronically or writing it down. Still, very few people can remember even 12 random words. This paper proposes an alternative recovery option that utilizes lower-than-standard entropy secrets (such as passwords, biometrics, and object extractors). It can be used on its own (in combination with strong key derivation functions) or provide an additional backup option for the existing mnemonics. In this work, we investigate several aspects of secure key derivation: (1) how much entropy different sources can provide; (2) what is the preferred construction of the fuzzy extractor; (3) our key contributions in the selected approach; (4) the main security assumptions and properties of Unforgettable Fuzzy Extractor; (5) economic rationale for parameters (e.g., fuzzy vault size, additional PoW difficulty, secret length, cost of the attack) achieving the optimal solution from both security and time perspectives.

Contents

1 Introduction

The best key recovery user experience — BIP-39 — has remained unchanged since 2013 [Pal+13]. In its essence, BIP-39 proposed a way to convert a computer-generated randomness into a human-readable transcription consisting of 12 words, which are surely more memorable and portable than a random 128-bit binary string. Despite its wide usage, the standard has a list of requirements and drawbacks:

- The input entropy must have a "good quality", which means that the user cannot choose it. Selecting preferred words from the dictionary in the order desired by the user provides a significant advantage to an attacker by reducing the entropy level, allowing them to reproduce the key more easily.
- Checksum defined in BIP-39 detects only random errors and does not protect from the list of targeted attacks (mnemonic substitution [qa20] or even pre-computed by the malicious wallet seed phrases [Led23]).
- Key derivation function is strongly defined to be `PBKDF2` [Kal00] with the iteration parameter of 2048. The standard does not support the use of more secure KDFs [BDK17] or increasing the iteration parameter to at least the level proposed by NIST [BDB10].
- *Over the shoulder problem.* If an attacker sees 12 written words in some storage, they always know that it is a mnemonic. If the mnemonic was not additionally protected with the passphrase, the seed can be recovered immediately. A passphrase increases the complexity of accessing the key by an observer, but it's comparable to password brute-forcing.
- If only several words from the seed phrase were lost, the user can brute-force them and recover the key. But in practice, users do not lose only a couple of words from the seed phrase; they lose the mnemonic as a whole.

There are other approaches to derive keys from more memorable data, such as passwords, biometrics, etc. But none of them are either secure or convenient for the end user. For instance, a random password of 20 characters can provide the user with 128 bits of entropy. However, the keyword in this sentence is "random." Typically, users *generate* come up with a familiar password, which leads to the need to increase their length to 50-60 characters to have enough entropy on the input. Brainwallets [Vas+17] is a similar interesting concept, which proposes selecting any data (a chapter in a book or a long sentence the user remembers) and passing it through the KDF function. This approach is better from the UX perspective (despite reentering the book chapter, not the simplest recovery process one can imagine), but security assumptions are much worse than for the randomly generated key. The use of biometrics as a single factor for accessing keys and funds is also not the best approach from the security perspective [BKR08].

In this study, we present **Unforgettable** – an approach that uses a diverse set of accessible sources (such as passwords, objects, or biometrics) to generate strong cryptographic keys. Unforgettable Fuzzy Extractor is aiming to resolve the biggest trilemma in the key management domain:

1. **Security**: derived keys must be strong, and there should not be any efficient approach for an attacker to reconstruct the key in a reasonable time.
2. **Usability**: key generation and recovery flows are supposed to be much simpler and friendlier, even compared to the BIP-39 approach.
3. **Recoverability**: the user should be able to recover their key in a reasonable time, even in case that some part of the used secret is lost.

Structure. In Section 2, we introduce all essential tools used throughout our study: notation, recognition neural networks, and basic security notions. In Section 3, we describe the central cryptographic primitive — Fuzzy Extractor. We define what it means for Fuzzy Extractor to be secure and specify previous constructions in the literature. Finally, in Section 4, we specify the *Goppa Code Offset* and *X-Lock* fuzzy extractor constructions used in Unforgettable. We introduce a modification of this approach that combines the stable source and several (unstable) biometric sources. In Section 4.2, we summarize the whole construction and discuss its properties. We reserve all technical details regarding security for Appendix.

2 Technical Preliminaries

2.1 Basic Notation

2.1.1 Metric Spaces

Throughout all the sections, we are going to work with various *metric spaces* (spaces equipped with a positive, symmetric distance function satisfying the triangle inequality). We equip spaces of form Σ^n over the finite alphabet Σ with the ***normalized*** *Hamming metric*:

$$\Delta(\mathbf{x}, \mathbf{y}) \triangleq \frac{1}{n} \cdot \#\{i \in [n] : x_i \neq y_i\}, \quad \text{with } \mathbf{x}, \mathbf{y} \in \Sigma^n,$$

where here and after by $[n]$ we denote the set $\{0, 1, \dots, n-1\}$. We call the distance from $\mathbf{x} \in \Sigma^n$ to the zero string by *Hamming weight* of $\mathbf{x}$ and denote by $\omega_H(\mathbf{x}) \triangleq \Delta(\mathbf{x}, \mathbf{0})$. In turn, when working over vector spaces of form $\mathbb{R}^n$, we typically use L^p distance, defined as

$$d_p(\mathbf{x}, \mathbf{y}) \triangleq \|\mathbf{x} - \mathbf{y}\|_p, \quad \|\mathbf{x}\|_p \triangleq \left(\sum_{i \in [n]} |x_i|^p \right)^{1/p}, \quad \text{with } \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

When writing $d(\mathbf{x}, \mathbf{y})$, we assume that we equip $\mathbb{R}^n$ with the Euclidean metric (i.e., we use L^2 distance), if not stated otherwise.

2.1.2 Bit-wise operations

As usual, $\mathbf{x} \oplus \mathbf{y}$ denotes the XOR operation over $\mathbf{x}, \mathbf{y} \in \{0,1\}^n$. Also, we will use $\bigoplus_{i=0}^{n-1} a_i = \bigoplus_{i \in [n]} a_i = a_0 \oplus \dots \oplus a_{n-1}$ as a XOR-ation operator over the set of indexes i. Given the arbitrary binary string $\mathbf{s}$ of length n, we write $\mathtt{MAJ}(\mathbf{s})$ to denote the *majority* function that outputs the most common bit in $\mathbf{s}$. We additionally introduce $\mathtt{MAJ}_\delta(\cdot)$ — a *hard majority function* that returns $\perp$ in case $|\omega_H(\mathbf{s}) - \frac{1}{2}| < \delta$ and $\mathtt{MAJ}(\mathbf{s})$ otherwise. In particular, $\mathtt{MAJ}_0 \equiv \mathtt{MAJ}$.

2.2 Error-Correction Codes

We denote the code over alphabet Σ by $\mathcal{C}$ and call it an (n, k, d) code if (1) $\mathcal{C} \subseteq \Sigma^n$, (2) messages are elements of Σ^k, and (3) distance of the code is d, where the distance of the code is defined as usual: $d(\mathcal{C}) \triangleq n \cdot \min_{\mathbf{x}, \mathbf{y} \in \mathcal{C}, \mathbf{x} \neq \mathbf{y}} \Delta(\mathbf{x}, \mathbf{y})$. We call $\mathcal{C}$ a *(q-ary) linear error-correction code* if $\Sigma = \mathbb{F}_q$ and $\mathcal{C}$ is a linear subspace of Σ^n with $\dim \mathcal{C} = k$. Finally, we always assume $d = 2t + 1$ where t is the *unique decoding distance*: that is, we assume our codes can detect up to $2t + 1$ errors and correct up to t errors, including.

Given the error-correcting code $\mathcal{C}$ over alphabet Σ, we denote encoding of a message $\mathbf{m} \in \Sigma^k$ by $\mathcal{C}.\mathsf{Enc}(\mathbf{m})$ that outputs the valid codeword. Similarly, given a word $\mathbf{z} \in \{0,1\}^n$, we denote the error-correction procedure as $\mathcal{C}.\mathsf{Dec}(\mathbf{z})$.

One important class of error-correction codes that gives parameters $(2^m, 2^m - mt, 2t+1)$ and that we further extensively rely on is the *Goppa Codes*.

Definition 2.1. *Consider the finite field $\mathbb{F}_{q^m}$ and a locator set $\mathcal{L} = \{\alpha_i\}_{i\in[n]} \subseteq \mathbb{F}_{q^m}$. Fix an irreducible polynomial $g \in \mathbb{F}_{q^m}[z]$ with $\deg g = t$.* ***Goppa code*** *Γ over $\mathcal{L}$ and g is defined as:*

$$\Gamma = \Gamma(\mathcal{L}, g(z)) \triangleq \left\{ \mathbf{c} = (c_0, \ldots, c_{n-1}) \in \mathbb{F}_q^n : \sum_{i\in[n]} \frac{c_i}{z - \alpha_i} = 0 \text{ in } \mathbb{F}_{q^m}[z]/\langle g \rangle \right\}$$

Correction procedure. Assume we need to decode received codeword $\mathbf{y} = (y_0, \ldots, y_{n-1})$ with r errors for $2r + 1 \leq d$. Suppose therefore $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \Gamma(\mathcal{L}, g)$ and $\omega_H(\mathbf{e}) = r$. To correct $\mathbf{y}$, we need to locate position of errors, say, $\mathcal{B} := \{i \in [n] : e_i \neq 0\}$, and determine the corresponding $\{e_i\}_{i\in\mathcal{B}}$. Define the *error-locator polynomial* $\sigma(z)$ and *error-evaluator polynomial* $w(z)$ as follows:

$$\sigma(z) \triangleq \prod_{i\in\mathcal{B}} (z - \alpha_i), \quad w(z) \triangleq \sum_{i\in\mathcal{B}} e_i \prod_{j\in\mathcal{B}\setminus\{i\}} (z - \alpha_j).$$

Finally, define the *syndrome* $S(\mathbf{y}) \triangleq \sum_{i\in[n]} \frac{y_i}{z-\alpha_i} \equiv \sum_{i\in\mathcal{B}} \frac{e_i}{z-\alpha_i} \pmod{g(z)}$.

Proposition 2.1. *Suppose $\mathbf{y} = \mathbf{c} + \mathbf{e}$ with $\mathbf{c} \in \Gamma(\mathcal{L}, g)$ and $r := \omega_H(\mathbf{e}) \leq [t/2]$. Then,*

- *Degrees of σ and w are as follows:* $\deg \sigma = r$, $\deg w \leq r - 1$.
- *σ and w are relatively prime: gcd$(\sigma(z), w(z)) = 1$.*
- *Error bits can be found as $e_k = w(\alpha_k)/\sigma'(\alpha_k)$ where $k \in \mathcal{B}$ and σ' is the formal derivative of $\sigma(z)$.*
- *One has $\sigma(z)S(\mathbf{y}) = w(z)$ modulo $g(z)$.*

The Proposition 2.1 gives rise to the following correction procedure.

Goppa Code Decoding Procedure $\Gamma.\mathsf{Dec}$

Input: $\mathbf{y} = (y_0, \ldots, y_{n-1}) \in \mathbb{F}_{q^m}$ with Goppa code $\Gamma(\mathcal{L} = \{\alpha_i\}_{i\in[n]}, g(z))$.
Algorithm:

1. Compute the syndrome $S(\mathbf{y}) = \sum_{i\in[n]} \frac{y_i}{z-\alpha_i}$.
2. Solve the equation $\sigma(z)S(\mathbf{y}) = w(z) \pmod{g(z)}$ with respect to coefficients of w and σ: namely, $\sigma(z) = z^r + \sum_{i\in[r]} \sigma_i z^i$ and $w(z) = \sum_{i\in[n]} w_i z^i$. If $q = 2$, take $w(z) := \sigma'(z)$.
3. Compute $\mathcal{B} = \{i \in [n] : \sigma(\alpha_i) = 0\}$ and compute errors $e_i = w(\alpha_i)/\sigma'(\alpha_i)$ for $i \in \mathcal{B}$.

Output: $\mathbf{c} = \mathbf{y} - \mathbf{e} \in \Gamma(\mathcal{L}, g)$.

Remark. While the aforementioned construction works over fields of arbitrary finite characteristic, we further always assume $q = 2$.

2.3 Object Recognition Neural Networks

2.3.1 Face Recognition Model

The typical problem setup in the face recognition research is as follows: given two facial images, say, $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{W \times H \times C}$, determine whether they correspond to the same person. Comparing two objects in the high-dimensional space (of dimensionality $W \cdot H \cdot C$) is vastly suboptimal. A more sufficient approach is therefore to train the feature extraction neural network $f : \mathbb{R}^{W \times H \times C} \to \mathbb{R}^d$ that projects the high-dimensional sample into a much lower-dimensional space $\mathbb{R}^d$, where d typically ranges from 128 to 1024.

What do we expect from such a projection? We want to obtain the following two properties: (1) given two images of the same person, say, $\mathbf{x}$ and $\mathbf{x}'$, the distance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ is *small*, while (2) given two images of two different people, say, $\mathbf{x}$ and $\mathbf{y}$, we expect that distance between $f(\mathbf{x})$ and $f(\mathbf{y})$ would be *large*. Of course, due to the probabilistic nature of neural networks, we cannot expect that $f(\mathbf{x})$ and $f(\mathbf{x}')$ would perfectly match. Instead, we define the *threshold* $\varepsilon \in \mathbb{R}_{>0}$ and consider images $\mathbf{x}$ and $\mathbf{x}'$ to be of the same person iff $d(f(\mathbf{x}), f(\mathbf{x}')) \leq \varepsilon$. Usually, we assume $d(\cdot, \cdot)$ is the L^2 metric.

So how do we train f? Some early approaches [TP91; BHK97], such as *Eigenface* or *Fisherface*, proposed to use the Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) methods to learn the linear embedding $f(\mathbf{x}) = \Pi\mathbf{x} + \boldsymbol{\beta}$. However, better results, obtained e.g. by [SKP15; Liu+17; Wan+18; Den+22], proposed to represent f as a neural network that outputs features on the unit d-dimensional hypersphere $S^{d-1} \triangleq \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$. While all the mentioned papers employ different yet similar methods, the general idea is to construct a loss function that reduces the intra-class average distance while increasing the inter-class distance. For instance, in *FaceNet* [SKP15], given the parameterized class of models $\mathcal{F}$ and a margin $\gamma \in \mathbb{R}_{\geq 0}$, one chooses the model that would maximize the following probability [War+21] over the distribution ρ of triplets of form $(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-)$ where $\mathbf{x}$ and $\mathbf{x}^+$ are images of the same person while $\mathbf{x}$ and $\mathbf{x}^-$ of different people:

$$\widehat{f} := \arg\max_{f \in \mathcal{F}} \Pr_{(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) \sim \rho}[\gamma + d(f(\mathbf{x}), f(\mathbf{x}^+)) < d(f(\mathbf{x}), f(\mathbf{x}^-))].$$

This is achieved by minimizing the expected value of the *triplet loss function*:

$$\mathcal{L}_{\text{triplet}}(f) \triangleq \mathbb{E}_{(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) \sim \rho}[\text{ReLU}(d(f(\mathbf{x}), f(\mathbf{x}^+)) - d(f(\mathbf{x}), f(\mathbf{x}^-)) + \gamma)]$$

Thus the model is chosen by $\widehat{f} \leftarrow \arg\min_{f \in \mathcal{F}} \mathcal{L}_{\text{triplet}}(f)$ by means of gradient descent.

2.3.2 Real Vector Quantization

As mentioned in the previous section, the neural network f outputs the d-dimensional real-valued vector. Unfortunately, for cryptographic applications, we cannot operate with the real-valued quantities. To resolve this issue, we map continuous values to the vector of elements from the finite alphabet Σ using what we call a *quantization method*. For simplicity, assume $\Sigma = \mathbb{Z}_r$ where r is the alphabet's size. Given the output of the neural network $\mathbf{v} = f(\mathbf{x}) \in S^{d-1}$, we map it to $q(\mathbf{v})$ where $q : S^{d-1} \to \mathbb{Z}_r^d$ is the *quantization function*. Below, we specify functions $q(\cdot)$ used to quantize the real-valued vector.

Binary Naive Embedding. This approach was used, in particular, in [KZF24]. In this method, the alphabet is binary, so $r = 2$. Upon receiving the feature vector $(v_1, \dots, v_d) \in S^{d-1}$, the output is the binary vector $(b_1, \dots, b_d) \in \{0,1\}^d$ where $b_i = \mathbb{1}[v_i \geq 0]$ with $\mathbb{1}[\cdot]$ being the indicator function. A slight modification of this approach is to estimate the average values of each feature vector component: say, $\boldsymbol{\mu} = (\mu_1, \dots, \mu_d)$, and form b_i as $b_i := \mathbb{1}[v_i \geq \mu_i]$.

Equal-probability quantization. This approach was used, in particular, in [Don+24]. This method partitions the range of feature values into r bins such that each bin has approximately an equal probability mass under the empirical feature distribution. More specifically, assume that ξ_i is the random variable representing the i^{th} component of the neural network output $\boldsymbol{\xi} = (\xi_1, \dots, \xi_d)$ with the marginal distribution $p_i(x)$. Partition the interval $[-1, 1]$ into q bins $\{(\tau_{i,j}, \tau_{i,j+1})\}_{j \in [r]}$ with the property that $\int_{(\tau_{i,j}, \tau_{i,j+1})} dp_i = \frac{1}{r}$ for each bin $(\tau_{i,j}, \tau_{i,j+1})$. Upon receiving the feature vector $(v_1, \dots, v_d) \in S^{d-1}$, we convert it to the quantized vector $(\widehat{v}_1, \dots, \widehat{v}_d) \in \mathbb{Z}_r^d$ such that $v_i \in (\tau_{i,\widehat{v}_i}, \tau_{i,\widehat{v}_i+1})$.

Random Projection. Another prominent approach for binary quantization ($r = 2$) thoroughly theoretically analyzed in [ZS21; DS18; YCP15] is surprisingly simple: quantize the feature vector $\boldsymbol{v} \in S^{d-1}$ by computing $\widehat{\boldsymbol{v}} := \mathsf{sign}(\Pi \boldsymbol{v})$ where $\Pi \in \mathbb{R}^{m \times d}$ is a random projection matrix generated by sampling each matrix component from the standard normal distribution: $\Pi_{i,j} \sim \mathcal{N}(0,1)$. In particular, [YCP15] proved the following theorem about such a mapping.

Theorem 2.1. *Consider an arbitrary finite set $K \subseteq S^{d-1}$ of (face) recognition model outputs, and let the output dimensionality $m \geq (c/\delta^2)|K|$. Let $\Delta(\cdot,\cdot)$ be the normalized Hamming distance, $d(\cdot,\cdot)$ the Euclidean distance, $\widehat{\cdot}$ is the quantized value using random projection $\widehat{\mathbf{x}} = \mathsf{sign}(\Pi \mathbf{x})$. Then, for all $\mathbf{x}, \mathbf{y} \in K$, we have:*

$$\Pr\left[|\Delta(\widehat{\mathbf{x}}, \widehat{\mathbf{y}}) - d(\mathbf{x}, \mathbf{y})| \leq \delta\right] \geq 1 - 2\exp(-\delta^2 m).$$

During our research, for each chosen face recognition model, we benchmark each of the quantization methods and select one that yields the smallest error. Additionally, to simplify notation, we always assume that the neural network f outputs the element from $\mathbb{Z}_r^d$ as the result of model inference and the subsequent quantization process.

2.3.3 Neural Network Accuracy Estimation

To further model the security level of the Unforgettable Extractor, we estimate the following parameters:

FRR & FAR (False Rejection Rate & False Acceptance Rate). Fix threshold $\varepsilon \in \mathbb{R}_{>0}$ and the neural network f. Assume ρ_{same} is the distribution of images of the same people, while ρ_{diff} is the distribution of images of different people. Then, we define FRR and FAR as the following quantities:

$$\mathrm{FRR}^f(\varepsilon) = \Pr_{(\mathbf{x},\mathbf{x}') \sim \rho_{\mathsf{same}}}[\Delta(f(\mathbf{x}), f(\mathbf{x}')) > \varepsilon], \quad \mathrm{FAR}^f(\varepsilon) = \Pr_{(\mathbf{x},\mathbf{y}) \sim \rho_{\mathsf{diff}}}[\Delta(f(\mathbf{x}), f(\mathbf{y})) \leq \varepsilon].$$

Empirically, we estimate these parameters as follows: fix the set of pairs of the same people from specifically chosen dataset $\mathcal{P}_{\mathsf{same}} \subseteq (\mathbb{R}^{W \times H \times C})^2$ and similarly of different people $\mathcal{P}_{\mathsf{diff}}$. Then,

$$\mathrm{FRR}^f(\varepsilon) \approx \frac{1}{|\mathcal{P}_{\mathsf{same}}|} \sum_{(\mathbf{x},\mathbf{x}') \in \mathcal{P}_{\mathsf{same}}} \mathbb{1}[\Delta(f(\mathbf{x}), f(\mathbf{x}')) > \varepsilon],$$

$$\mathrm{FAR}^f(\varepsilon) \approx \frac{1}{|\mathcal{P}_{\mathsf{diff}}|} \sum_{(\mathbf{x},\mathbf{y}) \in \mathcal{P}_{\mathsf{diff}}} \mathbb{1}[\Delta(f(\mathbf{x}), f(\mathbf{y})) \leq \varepsilon].$$

These two parameters are the crucial indicators of the neural network accuracy and security. The larger values of FRR result in a worse user experience, while larger FAR reduces the security level of the neural network. In fact, regardless of the further fuzzy extractor construction, the upper bound of the security of the single biometric source is roughly limited by $\log_2(1/\mathrm{FAR}^f(\varepsilon))$ bits where ε is the threshold below which two feature vectors map to the same secret value. Indeed, the adversary might grind through approximately $\log_2(1/\mathrm{FAR}^f(\varepsilon))$ biometric samples to get the matching feature vector. A slightly more rigorous justification is given in Theorem 2.2. Based on modern studies and our experiments, currently it is possible to achieve up to ≈ 20 bits of FAR while still having a FRR in the range 0.1–0.2.

Average Separation. To understand how much two feature vectors differ, given they come from the same or different people, we introduce *average separation* values as follows:

$$d^+(f) := \mathbb{E}_{(\mathbf{x},\mathbf{x}')\sim\rho_{\text{same}}}[\Delta(f(\mathbf{x}), f(\mathbf{x}'))], \quad d^-(f) := \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\rho_{\text{diff}}}[\Delta(f(\mathbf{x}), f(\mathbf{y}))].$$

They are estimated in a similar manner. These two values will be used to model the likelihood that two individual elements in the feature vector differ, thereby understanding the robustness and security of the fuzzy extractor. For an “ideal” neural network, we expect d^+ to be as close as possible to 0 while d^- to be approximately $\frac{1}{2}$. While the latter condition is typically observed in modern neural networks, the value of d^+ is in practice close to $\frac{1}{4}$, as our experiments show. For concrete distribution of distances, refer to Figure 1. In particular, similar results were obtained in [KZF24].

Datasets. For benchmarking models, we use LFW [Hua+07] and CelebA [Liu+15] datasets. These datasets are currently the most widely used in Face Recognition research and provide a comprehensive list of more than a million images with varying lighting conditions, races, or poses.

We provide benchmark results in Table 1.

Table 1: Modern neural networks accuracy metrics results based on LFW [Hua+07] and CelebA [Liu+15].

Recognition Model f	**Dataset**	**Quantization method**	$d^+(f)$	$d^-(f)$	FAR
InsightFace & ResNet100	LFW	Random Projection	0.254	0.498	2^{-21}
MagFace & iResNet50	LFW	Random Projection	0.231	0.482	2^{-20}
AdaFace & iResNet101	LFW	Random Projection	0.257	0.498	2^{-21}
Curricular & iResNet50	LFW	Equal-probability	0.249	0.498	2^{-21}

2.4 How Much Entropy Can We Extract From Different Sources?

2.4.1 Security Requirements

The security level is a measure of the complexity of attacking the corresponding cryptographic primitive. A highly simplified interpretation is that the system has λ-bit security if $\lambda = \log_2(T(\mathcal{A})/\epsilon(\mathcal{A}))$ where $T(\mathcal{A})$ is the running time of an attack while $\epsilon(\mathcal{A})$ is the success probability [MS24]. Such expression is minimized with respect to effective adversary strategies $\mathcal{A}$.

According to FIPS 140-2 [NIS01], the minimum acceptable security strength for cryptographic modules is 112 bits. At the same time, according to Aumasson [KKM19], we have the “broken” category (the primitive can be attacked) with $\lambda \leq 80$ and “wounded” with $80 < \lambda \leq 100$ (can be broken soon). So the minimal acceptable λ should be more than 100 (to make an attack currently impractical), and more than 112 to satisfy current security standards.

2.4.2 Sources and Entropy

Entropy is a measure of unpredictability or randomness. In practice, when we talk about entropy in the context of keys, we are asking: how many bits of real uncertainty are there?

For a discrete random variable X with probability distribution $p(x)$, the **entropy** $H(X)$ is defined as $H(X) := -\sum_x p(x)\log_2 p(x)$, where x ranges over possible outcomes of X [Sha48]. We also commonly want to estimate the lower bound on the entropy of the random variable. We estimate it using the **min-entropy** $H_\infty(X)$, defined as $H_\infty(X) := -\log_2 \max_{x\in\mathcal{X}} \Pr[X = x]$.

For randomly generated passwords, where we have a dictionary of q possible elements, the probability of any (uniformly chosen) password of length n is $p(x) = \frac{1}{q^n}$, so we have an entropy to be calculated as:

$$H = -\sum_x \frac{1}{q^n}\log_2\left(\frac{1}{q^n}\right) = n \cdot \log_2 q$$

If the password consists of printable ASCII characters, we have $H = n\cdot\log_2(94) \approx 6.55$ bits of entropy per character (assuming it is random). If the password consists of words from the BIP-39 dictionary, we have $H = n \cdot \log_2(2048) = 11$ bits per word.

When estimating entropy for biometrics and objects, we primarily consider FAR. The effective security per attempt can be calculated as $H_{\text{eff}} = -\log_2(\text{FAR})$ and $-\log_2(q \cdot \text{FAR})$ with q tries.

Theorem 2.2. *Let the verification system have a threshold ε, thus has a FAR value of $FAR^f(\varepsilon)$. Then the one-try attack min-entropy satisfies $H_\infty = -\log_2(FAR^f(\varepsilon))$. For q independent attempts at the same ε:*

$$H_\infty^q \geq -\log_2(1-(1-FAR^f(\varepsilon))^q) \geq -\log_2(\min\{1, q\cdot FAR^f(\varepsilon)\}$$

Proof. By the FIDO/ISO definition [Sch+24], FAR is exactly the success probability of a zero-effort impostor action at the chosen threshold, i.e., a one-trial attack success probability $H = -\log_2(p)$ (by the standard self-information/min-entropy mapping). Over q independent trials, this equates to $1-(1-p)^q$, which is $\leq qp$. So, the exact number of security bits can be calculated as $H_\infty^q = -\log_2(1-(1-\text{FAR}^f(\varepsilon))^q)$ or more conservatively $-\log_2(q\cdot\text{FAR}^f(\varepsilon))$.

We have the following estimates for different biometric sources in the Table 2.

Table 2: Number of security bits which can be retrieved from different sources

Source	Typical FAR range	Bits (up to)	References
Fingerprint	$10^{-4} - 10^{-6}$	20	[Wat+14; Mal+09]
Face	$10^{-4} - 10^{-6}$	20	[GNH22]
Iris	$10^{-7} - 10^{-12}$	40	[Dau03; QGM18]
Voice	$10^{-2} - 10^{-3}$	10	[KL10]
PUF	10^{-9}	30	[20]
Image's distinguishing points [1]	10^{-9}	30	raw estimations

When combining several independent sources $X_1,\ldots,X_n$ (e.g., several distinct biometric inputs), the total entropy is the sum of individual entropies: $H_\infty(X_1,\ldots,X_n) = \sum_{i=1}^n H_\infty(X_i)$, so intuitively we can retrieve enough security bits by combining different factors (sources) and using a secure fuzzy extractor.

[1]We are training and estimating a neural network that recognizes distinguishing points on pictures (book pages are a good source for this). We will provide practical results when they are ready.

3 Fuzzy Extractors

3.1 Definition

The vast majority of cryptographic authentication algorithms, one way or another, rely on the following rule: if the user knows (can reproduce) the *exact* secret value s, they are granted access to the system. The notion of *fuzzy extractors*, first introduced by Dodis [DRS04; Dod+08], allows us to define less strict rules where the user might know a close, yet not necessarily identical secret value s' such that if $\Delta(s, s')$ is small enough, the user can still access the system. As discussed previously, the secret value s is the embedding vector extracted from a source (e.g., a face or physical identifiers). Now, we define the *fuzzy extractor scheme* more formally and somewhat less generally than prior studies.

Further assume that each user's U biometric is associated with the corresponding feature vector distribution $\mathcal{W}_U$ over $\{0,1\}^d$.

Definition 3.1. *The **Fuzzy Extractor Scheme** Π over d-bit noisy biometric source, ℓ-bit secret values space, and v-bit public helper string space, is a tuple* $(\mathsf{Gen}, \mathsf{Rep})$ *that satisfy the following properties:*

- $\mathbf{Gen}(1^\lambda, \mathbf{w}) \to (\mathsf{hs}, \mathsf{sk})$. *Upon receiving the user's biometric sample $\mathbf{w} \in \{0,1\}^d$, the fuzzy extractor generates the public helper string $\mathsf{hs} \in \{0,1\}^v$ and secret value $\mathsf{sk} \in \{0,1\}^\ell$.*
- $\mathbf{Rep}(\mathbf{w}', \mathsf{hs}) \to \mathsf{sk}$. *Upon receiving the new user's biometric sample $\mathbf{w}' \in \{0,1\}^d$ and the helper data $\mathsf{hs} \in \{0,1\}^v$, the fuzzy extractor outputs the secret value $\mathsf{sk} \in \{0,1\}^\ell$.*

*We require the (δ, τ)-**correctness** property: for any biometric samples $\mathbf{w}, \mathbf{w}' \leftarrow\$ \mathcal{W}_U$ sampled from some user's distribution, the probability of successfully recovering the secret key is:*

$$Pr\left[\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) = \mathsf{sk} \;\middle|\; \begin{array}{c}(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ (\mathbf{w}, \mathbf{w}') \leftarrow\$ \mathcal{W}_U\end{array}\right] \geq 1 - \delta$$

Additionally, the probability of reproducing the same key using two biometric samples of different people: $(\mathbf{w}, \mathbf{w}') \leftarrow\$ \mathcal{W}_U \times \mathcal{W}_{U'}, U \neq U'$, is bounded above by another value which we call τ:

$$Pr\left[\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) = \mathsf{sk} \;\middle|\; \begin{array}{c}(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ (\mathbf{w}, \mathbf{w}') \leftarrow\$ \mathcal{W}_U \times \mathcal{W}_{U'}, U \neq U'\end{array}\right] \leq \tau$$

Now we need to define the security of the Fuzzy Extractor Scheme. Compared to standard cryptographic definitions (such as encryption or digital signatures), we cannot require negligible error for adversary strategies. Other than that, we will proceed as usual: define the attack game and define the security based on the adversary's probability of success.

Definition 3.2 (Secure Fuzzy Extractor). *For adversary algorithm $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$, define the following advantage value for any user U:*

$$SAdv[\mathcal{A}, \Pi] \triangleq 2 \cdot \left| Pr\left[b = \widehat{b} \;\middle|\; \begin{array}{l} \mathbf{w} \leftarrow\$ \mathcal{W}_U \\ (\mathsf{hs}, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}) \\ \mathsf{sk}_1 \leftarrow\$ \{0,1\}^\ell, b \leftarrow\$ \{0,1\} \\ \widehat{b} \leftarrow \mathcal{A}_1(\mathsf{hs}, \mathsf{sk}_b) \end{array}\right] - \frac{1}{2} \right|$$

We say that the Fuzzy Extractor Π is ***ε-secure*** *against secret value recovery if for any efficient adversary $\mathcal{A}$, one has SAdv$[\mathcal{A},\Pi] \leq \varepsilon$. Intuitively, the helper data* hs *reveals a small amount of information (namely, $\log_2 \frac{1}{\varepsilon}$ bits) about the secret* sk. *In particular, if $\varepsilon = \mu(\lambda)$ is negligible in λ, the Fuzzy Extractor scheme Π is computationally secure.*

Summary. This way, the fuzzy extractor scheme Π properties are determined by four parameters: $(\delta, \tau, \varepsilon)$, where the first parameter δ measures the false rejection rate, the second τ — false acceptance rate, and the third ε — security of Π.

3.2 Threat Model

We shall now discuss how to estimate the aforementioned Fuzzy Extractor parameters (*(δ, τ)-correctness* and *ε-security*) practically. Compared to security analysis of most classical cryptography primitives, where derivations are based on certain assumptions (such as the Discrete Log or Computational Diffie-Hellman assumptions), we will introduce several attack scenarios and test our definitions against these models. We propose the following three models:

Threat Model #1. The PPT adversary $\mathcal{A}_1$ views the helper string $\mathsf{hs} = (I, V)$ and tries to derive the correct secret entropy $\boldsymbol{\beta}$ corresponding to hs. The probability of doing this successfully gives the value of ε-security . However, during attack, $\mathcal{A}_1$ *does not access neural networks or any other input data (e.g., biometric).* This is the weakest assumption for basic protocol testing, and we typically expect ε to be negligible in this setting (or at least of the value of FAR).

Threat Model #2. The PPT adversary $\mathcal{A}_2$ views the helper string $\mathsf{hs} = (I, V)$ and without knowing the user's input, tries to achieve the same goals as in the Threat Model #1, but they are free to sample images from the distribution (e.g., take facial images from the dataset) and run the neural network over them. This is a quite natural and reasonable model, which provides one of the most optimal attack vectors.

Threat Model #3. The PPT adversary $\mathcal{A}_3$ is the same as in Threat Model #2, but they additionally know the identity of user possessing the helper string $\mathsf{hs} = (I, V)$ (*e.g.*, they know the user's facial photo).

Finally, we need to model the capabilities of the adversary to break our system. When working with stable cryptographic primitives such as digital signatures or hash functions, we either pass the verification or not. In case of biometric authentication, we have: (a) different probabilities of achieving proximity to the "correct" feature value, (b) different probabilities of passing verification with the given proximity. For that reason, we introduce Assumption 3.1, which states the capabilities of the adversary.

Proposition 3.1. *Assume, given the neural network f, the user's U biometrics distribution $\mathcal{W}_U$, and any biometric sample $\mathbf{w} \hookleftarrow\$ \mathcal{W}_U$ from this distribution, the adversary $\mathcal{A}$ can output the ε-close biometric sample $\mathbf{w}'$ (for $\varepsilon = k/n$ with $0 \leq k \leq n$) by calling the oracle $\mathcal{O}^f(\cdot)$ with probability $\rho^-(\varepsilon)$. More concretely,*

$$Pr\left[\Delta(\mathbf{w}, \mathbf{w}') = \varepsilon \;\middle|\; \begin{array}{c} \mathbf{w} \hookleftarrow\$ \mathcal{W}_U \\ \mathbf{w}' \leftarrow \mathcal{O}^f(\cdot) \end{array}\right] = \rho^-(\varepsilon)$$

In particular, cumulative distribution function (CDF) of $\rho^-(\varepsilon)$ is given by FAR$^f(\varepsilon)$.

This assumption is reasonable: if FAR$^f(\varepsilon)$ of the neural network f is, say, 2^{-20} (corresponding to modern face recognition neural networks), then by sampling 2^{20} biometric samples, the adversary is very likely to sample $\mathbf{w}'$ such that $\Delta(\mathbf{w}', \mathbf{w}) \leq \varepsilon$. In particular, it is reasonable to expect that the cumulative probability of getting distance below ε is $\sum_{k=0}^{\varepsilon n} \rho^-(k/n) \approx \text{FAR}^f(\varepsilon)$.

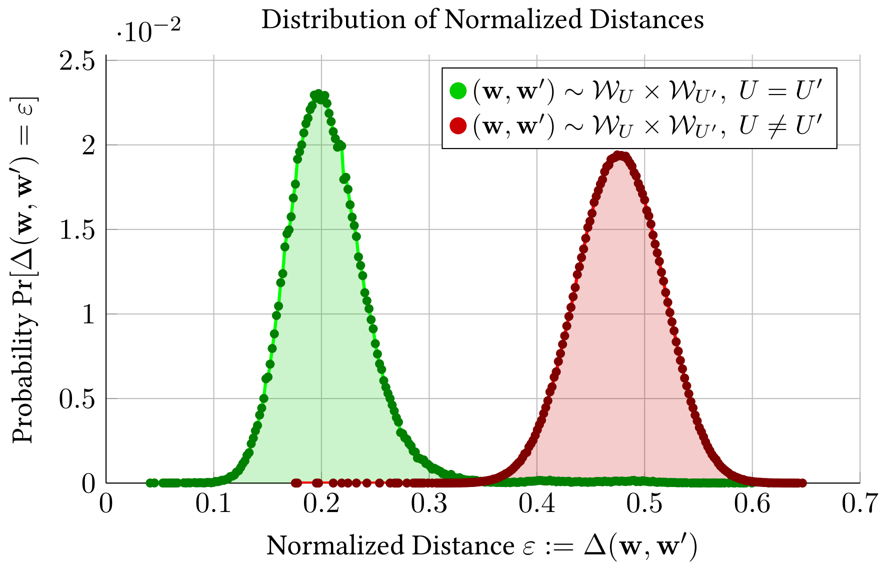

Figure 1: Distribution of distances: one distribution $\rho^{+}(\varepsilon)$ (marked in green) corresponds to probability of getting given distances between *same* people, while the other $\rho^{-}(\varepsilon)$ (marked in red) corresponds to distances between different people. As can be seen, proximity parameters (expected values of each distribution) are approximately $d^{+}(f) \approx 0.20$ and $d^{-}(f) \approx 0.47$, respectively.

In turn, for the honest user, we have the following assumption.

Proposition 3.2. *Assume, given the neural network f, the user's U biometric distribution $\mathcal{W}_U$, and any two biometric samples $\mathbf{w},\mathbf{w}' \xleftarrow{\$} \mathcal{W}_U$ from this distribution, the probability of them being ε-close is determined with probability $\rho^{+}(\varepsilon)$:*

$$Pr[\Delta(\mathbf{w},\mathbf{w}') = \varepsilon \mid \mathbf{w},\mathbf{w}' \xleftarrow{\$} \mathcal{W}_U] = \rho^{+}(\varepsilon).$$

In particular, CDF of $\rho^{+}(\varepsilon)$ is given by $TAR^{f}(\varepsilon)$ (true acceptance rate).

These two distributions estimated on the real data are depicted in Figure 1.

Based on the aforementioned assumptions and propositions, we show the following property of the original X-Lock construction.

3.3 Previous Constructions

3.3.1 Code-Based Fuzzy Extractor

Since their introduction, multiple constructions and variations of fuzzy extractors have been proposed. One interesting class of fuzzy extractors relies on error-correcting codes [JW99; HAD06], such as Reed-Solomon codes [RS60], BCH [BR60], or Goppa codes considered in Section 2.2. As an example of the concrete fuzzy extractor construction, consider the *code-offset construction*, originally introduced in [DRS04].

Definition 3.3 (Code-Offset Fuzzy Constructor)**.** *Fix (n,k,d) linear error-correction code $\mathcal{C} \subseteq \{0,1\}^n$ and a random oracle $\mathcal{H}: \{0,1\}^n \to \{0,1\}^\mu$. Then, the fuzzy extractor $\Pi_{\mathcal{C}}$ is constructed as follows:*

- $\mathsf{Gen}(1^\lambda, \mathbf{w} \in \{0,1\}^n) \to (\mathsf{hs} \in \{0,1\}^n, \mathsf{sk} \in \{0,1\}^\mu)$. *Generate a random message $\mathbf{m} \leftarrow\!\!\$ \{0,1\}^k$ and encode it to get a random codeword $\mathbf{c} \leftarrow \mathsf{Enc}(\mathbf{m})$. Form the helper data as $\mathsf{hs} := \mathbf{w} \oplus \mathbf{c}$ while the secret data is $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{c})$. Output $(\mathsf{hs}, \mathsf{sk})$.*
- $\mathsf{Rep}(\mathbf{w}' \in \{0,1\}^n, \mathsf{hs} \in \{0,1\}^n) \to \mathsf{sk}$. *Compute the corrupted codeword $\mathbf{c}' \leftarrow \mathsf{hs} \oplus \mathbf{w}'$. Decode the codeword to get $\mathbf{c}$ and output $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{c})$.*

Let us now consider some of the properties of such a construction. First, it is easy to see that if the biometric samples $\mathbf{w}, \mathbf{w}'$ are close enough (namely, less than t), we can always reproduce the valid key:

Lemma 3.1. *The function $\Pi_{\mathcal{C}}.\mathsf{Rep}(\mathbf{w}')$ always reproduces the valid key if $\Delta(\mathbf{w}, \mathbf{w}') \le t$.*

Reasoning. Notice that $\mathbf{c}' = \mathsf{hs} \oplus \mathbf{w}' = \mathbf{c} \oplus (\mathbf{w} \oplus \mathbf{w}')$. Therefore, $\Delta(\mathbf{c}', \mathbf{c}) = \Delta(\mathbf{w}, \mathbf{w}')$. Since by the condition in lemma we have $\Delta(\mathbf{w}, \mathbf{w}') \le t$, then $\Delta(\mathbf{c}', \mathbf{c}) \le t$, so a decoding of $\mathbf{c}'$ would yield $\mathbf{c}$.

However, for $(\mathbf{w}, \mathbf{w}') > t$, the decoding procedure becomes exponentially difficult. Consider the following theorem to illustrate this.

Theorem 3.1. *Probability that $\mathcal{C}.\mathsf{Dec}(\mathbf{c}')$ would succeed in decoding to some valid codeword is negligible (both in values of t and n)*

Proof. The code $\mathcal{C}$ contains 2^k valid codewords. The decoding procedure for $\mathbf{c}'$ succeeds if and only if $\Delta(\mathbf{c}, \mathbf{c}') \le t$ for some valid $\mathbf{c} \in \mathcal{C}$. Since there are 2^k valid codewords, the decoding would succeed only if $\mathbf{c}'$ happens to be in the t-ball of some codeword. The total volume of these balls is $2^k \cdot V(n,t)$ where $V(n,t)$ is the volume of the Hamming ball of radius t. Thus, the searched probability is:

$$\frac{2^k \cdot V(n,t)}{2^n} = 2^{k-n} \sum_{i=0}^{t} \binom{n}{i} \le t \cdot 2^{k-n} \binom{n}{t} \approx \frac{t \cdot 2^k}{t!} \cdot 2^{-n} n^t = \mathrm{negl}(t,n)$$

3.3.2 Fuzzy Vault

Another interesting approach, so-called **Fuzzy Vault**, was proposed by Juels and Sudan [JS02]. The fuzzy vault is based on Shamir's secret sharing scheme [Sha79] and was originally designed for unordered biometric feature sets. In this scheme, a secret is encoded as an evaluation of a polynomial of degree $t-1$. Polynomial is evaluated in all n features. The user who can reproduce t-of-n features can find the correct evaluations and interpolate the polynomial. Randomly generated garbage points obscure the vault against brute-force attacks. Specifically, the scheme is constructed as follows.

Definition 3.4 (Fuzzy Vault)**.** *Fix parameters (n,k,r,t) where n is the biometric sample size, r is the size of helper data, k is the secret size, and $t-1$ is the polynomial degree. Then, the fuzzy extractor $\Pi_{\mathcal{V}}$ over the Reed-Solomon code $\mathsf{RS}[\mathbb{F}_q, n, t]$ is constructed as follows.*

- $\mathsf{Gen}(1^\lambda, \mathbf{w} \in \mathbb{F}_q^n) \to (\mathsf{hs} \in \mathbb{F}_q^r, \mathsf{sk} \in \mathbb{F}_q^k)$. *Generate a random polynomial $f(X) \in \mathbb{F}_q^{<t}[X]$. Set secret value to $\mathsf{sk} \leftarrow \{f(\alpha)\}_{\alpha \in \mathcal{L}}$ where $\mathcal{L} \subseteq \mathbb{F}_q$ is the set of distinct field elements of size k (evaluation domain). Form the helper data as the set of points $\mathsf{hs} = \{(x_i, f(x_i))\}_{i \in [r]}$ where*

$x_i = w_i$ *for* $i \in [n]$ *and* $x_i \leftarrow\!\!\$\ \mathbb{F}_q$ *for all remaining indices* $i \in [r] \setminus [n]$. *Shuffle array* hs.

- **Rep**$(\mathbf{w}' \in \mathbb{F}_q^n, \mathsf{hs} \in \mathbb{F}_q^r) \to \mathsf{sk}$. *Parse the received helper data* hs *as* $\{(x_i, y_i)\}_{i \in [r]}$. *Form an array of points* $H := \{(x_i, y_i) \in \mathsf{hs} : x_i \approx w'_j \text{ for some } j \in [n]\}$[a]. *Run the Reed-Solomon decoding algorithm over* H *to get polynomial* $g \in \mathbb{F}^{<t}[X]$ *and output* $\mathsf{sk} \leftarrow \{g(\alpha)\}_{\alpha \in \mathcal{L}}$.

[a]Here, $x_i \approx w'_j$ can be either interpreted as the precise equality or, say, if $|x_i - w'_j|/q < \delta$ for some $\delta \in [0, 1]$.

Despite the promise of this approach, it introduces a significant challenge related to transforming features into field elements. While in Section 2.3.2 we described possible ways to construct such a map, the Fuzzy Vault construction must have a relatively large q (at least of characteristic n). This way, we would like to have a similar structure, but for the binary field. In the subsequent construction, we inherit *X-Lock Fuzzy Extractor* construction [Lib+24], and make a list of modifications increasing the complexity of attacks against an Unforgettable Fuzzy Extractor.

Other notable candidates. Other approaches include metric embedding techniques [DRS04] and specialized adaptations for biometric authentication [SLM07] and physical unclonable functions (PUFs) [MTV08]. These architectures have broadened the applicability of fuzzy extractors across secure storage, authentication, and key generation.

4 Unforgettable: Combining Multiple Sources and Approaches

4.1 Employed Fuzzy Extractors

In this section, we introduce two fuzzy extractor constructions that we employ in Unforgettable: the X-Lock, introduced in Section 4.1.1, and the Binary Goppa Code offset construction, discussed in Section 4.1.2. We briefly discuss their security levels and leave all the details in Appendix A.1.

4.1.1 X-Lock Construction

Unforgettable Fuzzy Extractor significantly relies on the construction of X-lock [Lib+24], which, compared to Fuzzy Extractor scheme with two methods, consists of three algorithms $\text{Init}(\cdot)$, $\text{Gen}(\cdot)$ and $\text{Rep}(\cdot)$. Our first slight modification is to merge the $\text{Init}(\cdot)$ and $\text{Gen}(\cdot)$ operations into a single one, thereby removing operational complexity and reducing the storage required for helper data. Let us see the construction.

Definition 4.1 (X-Lock Fuzzy Vault). *Suppose we have an* n*-bit biometric source,* k*-bit random indexing size,* μ*-bit secret entropy, and* ℓ *lockers per bit. Let* $\mathcal{H} : \{0,1\}^\mu \to \{0,1\}^\nu$ *be the random oracle. The* ***X-Lock Fuzzy Extractor*** *scheme* $\Pi_{\mathcal{X}}$ *consists of two procedures:*

- **Gen**$(1^\lambda, \mathbf{w} \in \{0,1\}^n) \to (\mathsf{hs}, \mathsf{sk})$. *Upon receiving the biometric sample* $\mathbf{w} \in \{0,1\}^n$*, the user generates the random secret entropy* $\boldsymbol{\beta} \leftarrow\!\!\$\ \{0,1\}^\mu$. *The user samples the* $\mu \times \ell$ *indexing matrix* I *which is formed as follows: each element is a set of indices* $\mathcal{I}_{i,j} \subseteq [n]$ *of length* k. *Finally, the user generates the vault matrix* $V \in \{0,1\}^{\mu \times \ell}$ *in which each element is formed as:*

$$V_{i,j} = L_{i,j} \oplus \beta_i, \quad L_{i,j} = \bigoplus_{\alpha \in \mathcal{I}_{i,j}} w_\alpha.$$

The public parameters are thus $\mathsf{hs} = (I, V)$ *while the secret key is* $\mathsf{sk} = \mathcal{H}(\boldsymbol{\beta})$. *We call the expression* $L_{i,j}$ *a locker, which XORs a random sub-vector of biometrics* $\mathbf{w}$.

- **Rep**$(\mathbf{w}' \in \{0,1\}^n, \mathsf{hs}) \to \mathsf{sk}$. *Upon receiving a new biometric sample* $\mathbf{w}' \in \{0,1\}^n$*, the user tries to recover* $\boldsymbol{\beta}$ *from* $\mathsf{hs} = (I, V)$ *as follows:*

$$\widehat{\beta}_i \leftarrow \mathtt{MAJ}\left\{\widehat{L}_{i,j} \oplus V_{i,j}\right\}_{j\in[\ell]}, \quad \widehat{L}_{i,j} = \bigoplus_{\alpha\in\mathcal{I}_{i,j}} w'_\alpha,$$

for each $i \in [\mu]$*. The user returns* $\mathsf{sk} \leftarrow \mathcal{H}(\widehat{\boldsymbol{\beta}})$.

We now give an intuitive explanation of what just happened. During the Gen$(\cdot)$ procedure, the user generates the secret entropy $\boldsymbol{\beta}$ and tries to conceal it using the biometric data $\mathbf{w}$. Moreover, instead of concealing $\boldsymbol{\beta}$ as a whole, the user conceals each bit β_i individually. This is done by choosing ℓ *lockers*, each of which is the XORation of $\mathbf{w}$ over the random set of indices. To restore the secret $\boldsymbol{\beta}$ using Rep$(\cdot)$ procedure, the user computes the new locker $\widehat{L}_{i,j} = \bigoplus_{\alpha\in\mathcal{I}_{i,j}} w'_\alpha$. For $\ell = 1$, the user is expected to restore the bit β_i iff $\widehat{L}_{i,1} = L_{i,1}$, or, in other words, XORation of the new biometric over the same set of indices results in the same bit value $L_{i,1}$. Realistically, we cannot expect that to happen for each $\{L_{i,1}\}_{i\in[\mu]}$, so instead we create multiple lockers per bit and hope that by choosing the most common value, the user will restore the bit correctly.

Proposition 4.1. *The Fuzzy Extractor scheme* $\Pi_\mathcal{X}$ *described in Definition 4.1, requires the public helper data of size approximately* $\mu\ell k \log n$ *bits and a secret key of size* ν *bits.*

Proof. The indexing matrix I consists of $\mu\ell$ entries, each of which is a collection of k elements from $[n]$. Thus, each element consumes up to $k \log n$ space, so the total size of I is $\mu\ell k \log n$. Vault matrix V in turns consists of $\mu\ell$ bits, which is drastically less than the size of I. Finally, since the output of the random oracle is an ν-bit string, the secret key is, unsurprisingly, a ν-bit string.

X-Lock Security

The construction of X-Lock is difficult to analyze due to its highly probabilistic nature. However, we outline the general intuition of its security and leave the details in Section A.1. Let us consider each of the parameters (n, μ, ℓ, k) and describe their influence on $(\delta, \tau, \varepsilon)$.

Parameter n. Since this is a feature vector size, this parameter is fixed. In our case, $n = 512$.

Parameter μ. The first trivial observation is that the entropy size μ is the first parameter that determines the security. Indeed, the simplest that an adversary can do (more specifically, in Threat Model #1) is to simply brute-force through possible values of the entropy $\boldsymbol{\beta}$. Thus, the security level of $\Pi_\mathcal{X}$ is at most μ bits. However, increasing μ also increases the false rejection rate, since more bits must be restored during the reproduction stage. Since we do not expect security to be significantly more than 20 bits (due to the neural network limitation), there is no point in making μ significantly larger than 60.

Remark. Since a neural network gives the security of 20 bits, it is tempting to set μ to 20–30. However, note that brute-forcing binary strings is much more computationally easy than brute-forcing by means of a face recognition model using various biometric samples. Although this does not influence the attack asymptotic, we should still keep this in mind.

Parameter ℓ. In turn, making ℓ larger decreases the false rejection rate. Moreover, it does not significantly influence the false acceptance rate, as our experiments and theoretical analysis show. However, this parameter dramatically influences the ε level: a larger ℓ makes attacking the X-Lock scheme much easier.

Here is how. If the adversary passes some (likely, incorrect) input $\mathbf{w}'$ to the $\mathsf{Rep}(\cdot)$ function, he can determine whether each β_i was recovered correctly (although this happens with a relatively low probability). Indeed, when computing $\{\widehat{L}_{i,j} \oplus V_{i,j}\}_{j\in[\ell]}$, the adversary can view the ratio of ones and zeros in the obtained binary string. If one gets a roughly 50/50 distribution, then the $\widehat{\beta}_i$ obtained was decoded incorrectly. In turn, if there is a clear majority, the majority value gives the correct value of β_i. Once β_i is known, the adversary gets ℓ equations of form $\bigoplus_{\alpha\in\mathcal{I}_{i,j}} w_\alpha = \beta_i \oplus V_{i,j}$ where $j \in [\ell]$. Now, here is the issue: suppose ℓ happens to be larger than n. Then, the number of unknowns in the equation (being clearly less than n) is less than the number of equations, thus the system, viewed as the linear equation over $\mathbb{Z}_2$, can be solved. In turn, assume $\ell \ll n$. Then, the number of unknowns $v = \left|\bigcup_{j\in[\ell]} \mathcal{I}_{i,j}\right|$ in the equation[2] is $k \leq v \leq \min\{n, k\ell\}$ with ℓ, again, being the number of equations. If $v \leq \ell$, the adversary recovers v bits of the feature vector, making subsequent attack easier. Otherwise, the system has $2^{v-\ell}$ solutions, and this is typically insufficient for the adversary to find any w_i. However, after unlocking enough bits β_i, the adversary can obtain enough equations to restore the whole binary string $\mathbf{w}$ (for instance, even n/ℓ bits of $\boldsymbol{\beta}$ might be sufficient).

Parameter k. From our theoretical analysis and original study [Lib+24], k is typically in range 3–8: picking larger k results in a too-large false rejection rate.

4.1.2 Binary Goppa Code Construction

As an alternative to the X-Lock construction, we employ the code-offset construction described in Section 3.3.1, equipped with the binary Goppa code Γ over $\mathbb{F}_{2^m}$. Since our neural network's outputs feature vectors of length 512, we set $m := 9$. This code has parameters $(2^m, 2^m - mt, 2t + 1)$. Compared to X-Lock construction, the security of the code-offset fuzzy extractor is much better understood. Indeed, for such a construction, as long as the new biometric sample is at a distance below t from the original one, the $\mathsf{Rep}(\cdot)$ procedure always succeeds, as we showed in Lemma 3.1 (for X-Lock, we can only compute the probability of decoding at specific distances). The following proposition gives the level of security we expect from this construction.

Theorem 4.1. *Consider the Goppa code Γ with parameters $(2^m, 2^m - mt, 2t + 1)$. As usual, denote the codeword size as $n := 2^m$. Then, we have the following parameters of the code-offset construction Π_Γ: $\delta = 1 - \mathit{TAR}^f(t/n)$, $\tau = \mathit{FAR}^f(t/n)$, $\varepsilon = \max\{\tau, 2^{-n+mt}\}$.*

Proof. Honest user decodes the secret value if the new biometric sample $\mathbf{w}'$ is at absolute Hamming distance below t. According to Assumption 3.2, if the user used $\mathbf{w}$ during $\mathsf{Gen}(\cdot)$ procedure, $\Pr[\Delta(\mathbf{w}, \mathbf{w}') \leq t/n] = \sum_{\varepsilon\leq t/n} \rho^+(\varepsilon)$. By definition, this is nothing but $\mathrm{TAR}^f(t/n)$. Thus $\delta = 1 - \mathrm{TAR}^f(t/n)$. The value of τ is obtained similarly, but we use the distribution $\rho^-(\cdot)$ instead.

To find ε, we consider two threat models, introduced in Section 3.2. Using Threat Model #1, the adversary $\mathcal{A}_1$ can simply brute-force through the message space, which consists of 2^k words. Since $k = n - mt$, the successful attack probability ε is at least $2^{-k} = 2^{-n+mt}$. Using Threat Model #2, the adversary can simply use $\mathcal{O}^f(\cdot)$ oracle and with probability $\mathrm{FAR}^f(t/n)$, find the feature vector (t/n)-close to $\mathbf{w}$. Thus, ε is the largest among the two probabilities.

Remark. Threat Model #3, in principle, does not differ from Threat Model #2 in terms of theoretical analysis. The only difference is the different probability when calling $\mathcal{O}^f(\cdot)$ oracle.

[2]In fact, the expected value of number of unknows is $\mathbb{E}[v] = n(1 - (1 - k/n)^\ell) \approx n(1 - e^{-\ell k/n})$.

4.2 Unforgettable Key Derivation

In our previous discussion, we have not yet defined how we combine multiple (both biometric and stable) sources. In this section, we focus on aggregating multiple sources.

4.2.1 Keyed Fuzzy Extractor

As discussed earlier, in the classical construction of X-Lock, during the recovery procedure for each β_i, based on the distribution of bits $\{\widehat{L}_{i,j} \oplus V_{i,j}\}_{j\in[\ell]}$, the adversary can distinguish whether he has correctly restored the value or not. Our proposal to partially address this issue is as follows: assume one has a relatively high-entropy, stable source, but it is insufficient to meet the requirements of modern standards (e.g., a 60-bit short password). We can then additionally blind each locker bit with the sample from this stable source. Assume the user knows the λ-bit password in addition to the used features. Knowing password $\mathbf{p} \in \{0,1\}^\lambda$ (for example, the password or passphrase passed through the KDF function) and fuzzy sample $\mathbf{w} \in \{0,1\}^n$, the values in the final vault are now constructed as:

$$V_{i,j} = L_{i,j} \oplus \beta_i, \quad L_{i,j} := \bigoplus_{\alpha\in\mathcal{I}_{i,j}} w_\alpha \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma,$$

where indices $\mathcal{J}_{i,j} \subseteq [\lambda]$ range over password bits. This construction ensures that, for an attacker, the resulting structure is uniformly random, given no knowledge of $\mathbf{p}$ and that $\mathbf{p}$ is distributed uniformly.

This motivates us to generalize this construction and define the *keyed fuzzy extractor*.

Definition 4.2 (Keyed Fuzzy Extractor). *The* ***Keyed*** *Fuzzy Extractor Scheme* Π *over* d*-bit noisy biometric source,* ℓ*-bit secret values space, and* v*-bit public helper string, is a tuple* $(\mathsf{Gen}, \mathsf{Rep})$ *that satisfy the following properties:*

- $\mathbf{Gen}(1^\lambda, \mathbf{w}, \mathbf{p}) \to (\mathsf{hs}, \mathsf{sk})$. *Upon receiving the user's biometric sample* $\mathbf{w} \in \{0,1\}^d$ *and stable password* $\mathbf{p} \in \{0,1\}^\nu$, Π *generates helper string* $\mathsf{hs} \in \{0,1\}^v$ *and secret value* $\mathsf{sk} \in \{0,1\}^\ell$.
- $\mathbf{Rep}(\mathbf{w}', \mathbf{p}', \mathsf{hs}) \to \mathsf{sk}$. *Upon receiving the new biometric sample* $\mathbf{w}'$ *and password* $\mathbf{p}'$*, the fuzzy extractor outputs the secret value* $\mathsf{sk} \in \{0,1\}^\ell$.

Similarly to the regular Fuzzy Extractor construction, we define the δ*-correctness similarly:*

$$Pr\left[\mathsf{Rep}(\mathbf{w}', \mathbf{p}', \mathsf{hs}) = \mathsf{sk} \;\middle|\; \begin{array}{l}(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}, \mathbf{p}) \\ (\mathbf{w}, \mathbf{w}') \leftarrow\!\!\$\; \mathcal{W}_U,\ \mathbf{p}' = \mathbf{p}\end{array}\right] \geq 1 - \delta$$

In turn, the value of τ *no longer makes sense: we require that for* $\mathbf{p} \neq \mathbf{p}'$*, upon calling the reproduction function* $\mathsf{sk}' \leftarrow \mathsf{Rep}(\mathbf{w}', \mathbf{p}', \mathsf{hs})$ *with* $(\mathsf{hs}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathbf{w}, \mathbf{p})$*, the probability of* $\mathsf{sk}' = \mathsf{sk}$ *is* $\leq$ *negl*(ν).

Similarly, we can easily extend the code-based construction. Indeed, instead of forming the helper data as $\mathsf{hs} := \mathbf{w} \oplus \mathbf{c}$, one can define $\mathsf{hs} := \mathbf{w} \oplus \mathbf{p} \oplus \mathbf{c}$ (note that in such case, the low-bit $\mathbf{p}$ might need to be extended to n bits). Or, even simpler, define $\mathsf{hs} := (\mathbf{w} \parallel \mathbf{p}) \oplus \mathbf{c}$. It is then straightforward to check security for this construction: for the first case, δ remains almost the same since $\mathbf{p} \oplus \mathbf{p}'$ cancels out as soon as $\mathbf{p} = \mathbf{p}'$. Without knowing $\mathbf{p}$, in turn, the distance will be at least $\nu/2$, making decoding infeasible. Similar thoughts are true for the latter case as well.

4.2.2 Combining Multiple Sources

We can modify the previous construction further to get *Aggregated Fuzzy Extractor* by aggregating biometric samples $\{\mathbf{w}_s\}_{s\in[N]}$ from N different sources (say, face/voice/fingerprint/etc.). For X-Lock, this can be done by simply XORing all these values. This way, the final vault elements will be evaluated as:

$$V_{i,j} = L_{i,j} \oplus \beta_i, \quad L_{i,j} := \bigoplus_{s\in[N]} \bigoplus_{\alpha\in\mathcal{I}_{i,j}^{(s)}} w_{s,\alpha} \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma$$

We summarize the construction in Definition 4.3.

Definition 4.3 (Brain-Bio Fuzzy Extractor). *Suppose we have N biometric sources of bit-length n, indexing size k, μ-bit entropy, λ-bit stable source, and ℓ lockers per bit. Let $\mathcal{H} : \{0,1\}^\mu \to \{0,1\}^\nu$ be the random oracle. The **Brain-Bio Fuzzy Extractor** Π_B consists of two procedures:*

- **Gen**$(1^\lambda, \{\mathbf{w}_s\}_{s\in[N]} \subseteq \{0,1\}^n, \mathbf{p} \in \{0,1\}^\lambda) \to (\mathsf{hs}, \mathsf{sk})$. *Upon receiving N unstable biometric samples $\{\mathbf{w}_s\}_{s\in[N]} \subseteq \{0,1\}^n$ and stable input $\mathbf{p} \in \{0,1\}^\lambda$, the user generates the random secret entropy $\boldsymbol{\beta} \leftarrow\$ \{0,1\}^\mu$ and samples $N+1$ indexing matrices $(\{I^{(s)}\}_{s\in[N]}, J)$ of size $\mu \times \ell$, where each element $\mathcal{I}_{i,j}^{(s)}, \mathcal{J}_{i,j} \subseteq [n]$ is a set of indices of length k. The user generates the vault matrix $V \in \{0,1\}^{\mu\times\ell}$ in which each element is formed as:*

$$V_{i,j} = L_{i,j} \oplus \beta_i, \quad L_{i,j} = \bigoplus_{s\in[N]} \bigoplus_{\alpha\in\mathcal{I}_{i,j}^{(s)}} w_{s,\alpha} \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma$$

  The public parameters are then set to $\mathsf{hs} = (\{I^{(s)}\}_{s\in[N]}, J, V)$ while the secret key is $\mathsf{sk} = \mathcal{H}(\boldsymbol{\beta})$.

- ***Rep***$(\{\mathbf{w}'_s\}_{s\in[N]} \subseteq \{0,1\}^n, \mathbf{p} \in \{0,1\}^\lambda, \mathsf{hs}) \to \mathsf{sk}$. *Upon receiving new biometric samples $\{\mathbf{w}'_s\}_{s\in[N]} \subseteq \{0,1\}^n$, the user tries to recover $\boldsymbol{\beta}$ from $\mathsf{hs} = (\{I^{(s)}\}_{s\in[N]}, J, V)$ as follows:*

$$\widehat{\beta}_i \leftarrow \mathtt{MAJ}\left\{\widehat{L}_{i,j} \oplus V_{i,j}\right\}_{j\in[\ell]}, \quad \widehat{L}_{i,j} = \bigoplus_{s\in[N]} \bigoplus_{\alpha\in\mathcal{I}_{i,j}^{(s)}} w'_{s,\alpha} \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma$$

  for each $i \in [\mu]$. The user returns $\mathsf{sk} \leftarrow \mathcal{H}(\widehat{\boldsymbol{\beta}})$

Similarly, code-based construction can be extended using a similar idea.

Definition 4.4 (Aggregated Code-Based Construction). *For simplicity, assume one has N biometric sources of the same size n. Assume the password is of size ν. Suppose Γ is the Goppa code over $(nN+\nu)$-bit codewords with appropriate distance parameter. Define aggregated code-offset fuzzy extractor scheme Π_Γ:*

- **Gen**$(1^\lambda, \{\mathbf{w}_s\}_{s\in[N]}, \mathbf{p}) \to (\mathsf{hs}, \mathsf{sk})$. *Upon receiving N unstable biometric samples $\{\mathbf{w}_s\}_{s\in[N]} \subseteq \{0,1\}^n$ and stable input $\mathbf{p} \in \{0,1\}^\nu$, the user generates a random codeword $\mathbf{c} \leftarrow\$ \Gamma$. Secret key is then $\mathsf{sk} \leftarrow \mathcal{H}(\mathbf{c})$ and the helper data is $\mathsf{hs} \leftarrow (\mathbf{w}_0 \parallel \cdots \parallel \mathbf{w}_{N-1} \parallel \mathbf{p}) \oplus \mathbf{c}$.*

- **Rep**$(\{\mathbf{w}'_s\}_{s\in[N]}, \mathbf{p}', \mathsf{hs}) \to \mathsf{sk}$. *Output* $\Gamma.\mathsf{Dec}(\mathsf{hs} \oplus (\mathbf{w}'_0 \parallel \cdots \parallel \mathbf{w}'_{N-1} \parallel \mathbf{p}'))$.

4.3 X-Lock Global Fuzzy Registry

We presume the use of a single global **registry** with mixed vaults for all users. In this case, each user contributes to the security of the whole quorum and does not require an additional addressing service to have a context about the account they want to recover. The registry consists of one large matrix $\mathbf{V}$ and N matrices[3] $\{\mathbf{I}_s\}_{s\in[N]}$ of size $M \times \ell$ with N being the number of fuzzy sources for recovery, and M is the total vault size. If the user does not have enough factors to use, appropriate features for the missing ones can be set as 0^n, which does not affect the evaluation of the final fuzzy extractor elements.

As we mentioned earlier, the transition from BIP-39 to other approaches is not supposed to be instantaneous. Therefore, we propose an approach that enables us to create a seed phrase backup using Unforgettable Fuzzy Extractor. An entire flow is following:

Vault creation:

1. Sample a random entropy string: $\boldsymbol{\beta} \leftarrow\$ \{0,1\}^{\mu}$.
2. Select N factors and derive vectors of features from them: $\{\mathbf{w}_i\}_{i\in[N]} \subseteq \{0,1\}^n$. *We recommend using a password as one of the factors, the vector of features for which is calculated as* $\mathbf{p} = \mathcal{H}(\textsf{password})$. If some factors are missed, their features' vectors are equal to 0^n.
3. Generate N local index matrices $\{I^{(s)}\}_{s\in[N]}$ with elements $I^{(s)} = \{\mathcal{I}^{(s)}_{i,j}\}_{i\in[\mu],j\in[\ell]}$ with $I_{i,j} \in [n]$, each of a size $\mu \times \ell$.
4. Calculate the local vault $V \in \{0,1\}^{\mu\times\ell}$:
$$V_{i,j} = L_{i,j} \oplus \beta_i, \quad L_{i,j} = \bigoplus_{s\in[N]} \bigoplus_{\alpha\in\mathcal{I}^{(s)}_{i,j}} w_{s,\alpha} \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma.$$
5. Select μ random indices $J \subseteq [M]$ (such that $|J| = \mu$).
6. Insert local V's and I's rows into global vault positions defined by J:
$$\mathbf{V}[J_i] \leftarrow V_i, \quad \mathbf{I}_j[J_i] \leftarrow I^{(j)}_i, \quad i \in [\mu],\ j \in [N],$$
where V_i and $I^{(j)}_i$ denotes the ith row in corresponding matrices.
7. Set the seed key as: $\textsf{seed} = \text{kdf}(\boldsymbol{\beta})$. For concrete KDF function choice, refer to Section 4.3.1.

Key recovery:

1. Generate new fuzzy samples $\{\mathbf{w}'_i\}_{i\in[N]} \subseteq \{0,1\}^n$ and derive exactly the same password $\mathbf{p} = \mathcal{H}(\textsf{password})$.
2. Compute new locker bits $\widehat{L}_{i,j} \leftarrow \bigoplus_{s\in[N]} \bigoplus_{\alpha\in\mathcal{I}^{(s)}_{i,j}} w'_{s,\alpha} \oplus \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma$.
3. Restore entropy bits as: $\widehat{\beta}_i \leftarrow \texttt{MAJ}_\delta\{\widehat{L}_{i,j} \oplus \mathbf{V}_{i,j}\}_{i\in[M]}$.
4. Derive the seed in the same manner as in step 7 in the **vault creation** algorithm.

[3] In our first implementation, we use $N = 6$.

4.3.1 Key Derivation Function Choice

We use the following seed generation algorithm: seed $= \texttt{PBKDF2}(\boldsymbol{\beta}, \mathsf{mnemonic}||\mathsf{passphrase}, 2048, 512)$ if BIP-39 needs to be supported or seed $= \texttt{Argon2id}(\boldsymbol{\beta}||\mathsf{PoW}, \mathsf{params})$, otherwise.

5 Conclusions and Future Work

In this paper, we presented an Unforgettable Fuzzy Extractor — a key derivation mechanism that operates with memorable data (such as biometrics, passwords, or objects). We analyzed basic efficiency and security properties: (δ, τ)-correctness and ε-security. Additionally, we described a method to make Unforgettable Fuzzy Extractor compatible with existing standards (like BIP-39) and an approach to create a single fuzzy vault, which (1) allows for increasing the complexity of the selective attack to the vault, and (2) removes the need to have an identifier of the key/account that needs to be recovered.

The construction of the Unforgettable Fuzzy Extractor enables the combination with any preferred entropy source and neural network, allowing for the selection of optimal vault parameters based on FAR/FRR. Initially, it was designed due to the limited computational and recognition abilities of user devices, but it can also provide a significant advantage in cases where more accurate sensor usage is employed (for fingerprints, irises, heartbeats, etc.). Using XOR as a basic operation allows us to implement a hardware extractor easily as an extension to the highly sensitive sensors. The single concern relates to the required storage size, especially in the case of a single vault (even though the reproduction procedure can be efficiently accelerated through parallelization, running it directly on sensors might be a challenge).

We have several vectors and ideas for the following research. Firstly, it involves integrating the extractor with highly accurate sensors and empirical research on the vault size. Secondly, we aim to enhance the accuracy of models operating on mobile users' devices (for example, by building 3D models of scanned objects); the space of arbitrary objects is significantly larger than the space of biometric variants, so potentially a much larger number of security bits can be extracted. The third direction involves building models that retrieve security bits not from the physical features, but rather based on a context received during the interaction process with the user. The fourth direction involves adding privacy to the network layer, making the rows of a specific user's vault more indistinguishable from fake evaluations and those of other users.

Acknowledgments

We thank Pavel Kravchenko, Mykhailo Velykodniy, Yaroslav Panasenko, and Yevhen Hrubian for their kind review and feedback. A massive shoutout to the entire Rarimo and Distributed Lab teams for implementing and integrating the extractor proposed in the paper in a real-world application.

References

[Sha48] Claude E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x. URL: https://doi.org/10.1002/j.1538-7305.1948.tb01338.x.

[BR60] R. C. Bose and D. K. Ray-Chaudhuri. "On a Class of Error Correcting Binary Group Codes". In: *Information and Control* 3.1 (1960), pp. 68–79. DOI: 10.1016/S0019-9958(60)90287-4.

[RS60] Irving S. Reed and Gustave Solomon. “Polynomial Codes over Certain Finite Fields”. In: *Journal of the Society for Industrial and Applied Mathematics* 8.2 (1960), pp. 300–304. DOI: 10.1137/0108018.

[Sha79] Adi Shamir. “How to share a secret”. In: *Commun. ACM* 22.11 (Nov. 1979), pp. 612–613. ISSN: 0001-0782. DOI: 10.1145/359168.359176. URL: https://doi.org/10.1145/359168.359176.

[TP91] M.A. Turk and A.P. Pentland. “Face recognition using eigenfaces”. In: *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* 1991, pp. 586–591. DOI: 10.1109/CVPR.1991.139758.

[BHK97] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman. “Eigenfaces vs. Fisherfaces: recognition using class specific linear projection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19.7 (1997), pp. 711–720. DOI: 10.1109/34.598228.

[JW99] Ari Juels and Martin Wattenberg. “A Fuzzy Commitment Scheme”. In: *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS).* ACM, 1999, pp. 28–36. DOI: 10.1145/319709.319714.

[Kal00] Burt Kaliski. *PKCS #5: Password-Based Cryptography Specification Version 2.0.* RFC RFC 2898. Request for Comments 2898. RSA Laboratories, 2000. URL: https://www.rfc-editor.org/rfc/rfc2898.

[NIS01] NIST. *FIPS PUB 140-2: Security Requirements for Cryptographic Modules.* Tech. rep. FIPS PUB 140-2. https://csrc.nist.gov/publications/detail/fips/140/2/final. National Institute of Standards and Technology, 2001.

[JS02] Ari Juels and Madhu Sudan. “A Fuzzy Vault Scheme”. In: *IEEE International Symposium on Information Theory (ISIT).* IEEE, 2002, p. 408.

[Dau03] John Daugman. “The Importance of Being Random: Statistical Principles of Iris Recognition”. In: *Pattern Recognition* 36.2 (2003), pp. 279–291. DOI: 10.1016/S0031-3203(02)00030-4.

[DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”. In: *Advances in Cryptology – EUROCRYPT 2004.* Springer, 2004, pp. 523–540. DOI: 10.1007/978-3-540-24676-3_31.

[HAD06] Feng Hao, Ross Anderson, and John Daugman. “Combining Crypto with Biometrics Effectively”. In: *IEEE Transactions on Computers* 55.9 (2006), pp. 1081–1088. DOI: 10.1109/TC.2006.138.

[Hua+07] Gary B. Huang et al. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments.* Tech. rep. 07-49. University of Massachusetts, Amherst, Oct. 2007.

[SLM07] Y. Sutcu, Qiming Li, and Nasir Memon. “Protecting Biometric Templates with Sketch: Theory and Practice”. In: *IEEE Transactions on Information Forensics and Security* 2.3 (2007), pp. 503–512. DOI: 10.1109/TIFS.2007.902019.

[BKR08] Lucas Ballard, Seny Kamara, and Michael K Reiter. “The Practical Subtleties of Biometric Key Generation.” In: *USENIX Security Symposium.* 2008, pp. 61–74.

[Dod+08] Yevgeniy Dodis et al. “Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data”. In: *SIAM Journal on Computing* 38.1 (2008), pp. 97–139. DOI: 10.1137/060651380.

[MTV08] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. "A Soft Decision Helper Data Algorithm for SRAM PUFs". In: *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2008, pp. 2101–2105. DOI: 10.1109/ISIT.2008.4595271.

[Mal+09] Davide Maltoni et al. *Handbook of Fingerprint Recognition*. 2nd ed. Springer, 2009. DOI: 10.1007/978-1-84882-254-2.

[BDB10] William E. Burr, Donna F. Dodson, and Elaine M. Barker. *Recommendation for Password-Based Key Derivation: Part 1: Storage Applications*. Tech. rep. NIST SP 800-132. NIST Special Publication 800-132. National Institute of Standards and Technology (NIST), Dec. 2010. URL: https://csrc.nist.gov/publications/detail/sp/800-132/final.

[KL10] Tomi Kinnunen and Haizhou Li. "An Overview of Text-Independent Speaker Recognition: From Features to Supervectors". In: *Speech Communication* 52.1 (2010), pp. 12–40. DOI: 10.1016/j.specom.2009.08.009.

[Pal+13] Marek Palatinus et al. *BIP-39: Mnemonic code for generating deterministic keys*. https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki. Accessed: 2025-08-20. 2013.

[Wat+14] Craig I. Watson et al. *Fingerprint Vendor Technology Evaluation*. NIST Interagency/Internal Reports (NISTIR) 8034. National Institute of Standards and Technology (NIST), 2014. DOI: 10.6028/NIST.IR.8034. URL: https://doi.org/10.6028/NIST.IR.8034.

[Liu+15] Ziwei Liu et al. "Deep Learning Face Attributes in the Wild". In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A unified embedding for face recognition and clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2015, pp. 815–823. DOI: 10.1109/cvpr.2015.7298682. URL: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[YCP15] Xinyang Yi, Constantine Caramanis, and Eric Price. "Binary embedding: Fundamental limits and fast algorithm". In: *International Conference on Machine Learning*. PMLR. 2015, pp. 2162–2170.

[BDK17] Alex Biryukov, Daniel Dinu, and Dmitry Khovratovich. "Argon2: The Memory-Hard Function for Password Hashing and Other Applications". In: *IACR Transactions on Symmetric Cryptology* 2017.4 (2017), pp. 43–57. DOI: 10.13154/tosc.v2017.i4.43-57. URL: https://tosc.iacr.org/index.php/ToSC/article/view/729.

[Liu+17] Weiyang Liu et al. "Sphereface: Deep hypersphere embedding for face recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 212–220.

[Vas+17] Marie Vasek et al. "The Bitcoin Brain Drain: Examining the Use and Abuse of Bitcoin Brain Wallets". In: May 2017, pp. 609–618. ISBN: 978-3-662-54969-8. DOI: 10.1007/978-3-662-54970-4_36.

[DS18] Sjoerd Dirksen and Alexander Stollenwerk. "Fast binary embeddings with gaussian circulant matrices: improved bounds". In: *Discrete & Computational Geometry* 60.3 (2018), pp. 599–626.

[QGM18] George W. Quinn, Patrick Grother, and James Matey. *IREX IX Part One: Performance of Iris Recognition Algorithms*. NIST Interagency/Internal Reports (NISTIR) 8207. National Institute of Standards and Technology (NIST), 2018. DOI: 10.6028/NIST.IR.8207. URL: https://doi.org/10.6028/NIST.IR.8207.

[Wan+18] Hao Wang et al. “Cosface: Large margin cosine loss for deep face recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2018, pp. 5265–5274.

[KKM19] Kevin Kelly, Neal Koblitz, and Alfred Menezes. “The Joy of Factoring and the Factoring of Joy”. In: *IACR Cryptology ePrint Archive* 2019/1492 (2019). URL: https://eprint.iacr.org/2019/1492.

[qa20] Michael Folkson (question) and Kalle Rosenbaum (accepted answer). *Should the BIP 39 mnemonic sentence checksum be eliminated from the standard? Does it do more harm than good?* Bitcoin Stack Exchange, Question #100376, accessed 20 August 2025. Dec. 2020. URL: https://bitcoin.stackexchange.com/questions/100376/should-the-bip-39-mnemonic-sentence-checksum-be-eliminated-from-the-standard-do.

[20] *Information security, cybersecurity and privacy protection — Physically unclonable functions — Part 1: Security requirements.* ISO/IEC, 2020. URL: https://www.iso.org/standard/76353.html.

[War+21] Frederik Warburg et al. “Bayesian triplet loss: Uncertainty quantification in image retrieval”. In: *Proceedings of the IEEE/CVF International conference on Computer Vision.* 2021, pp. 12158–12168.

[ZS21] Jinjie Zhang and Rayan Saab. “Faster Binary Embeddings for Preserving Euclidean Distances”. In: *International Conference on Learning Representations.* 2021. URL: https://openreview.net/forum?id=YCXrx6rRCXO.

[Den+22] Jiankang Deng et al. “ArcFace: Additive Angular Margin Loss for Deep Face Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.10 (Oct. 2022), pp. 5962–5979. ISSN: 1939-3539. DOI: 10.1109/tpami.2021.3087709. URL: http://dx.doi.org/10.1109/TPAMI.2021.3087709.

[GNH22] Patrick Grother, Mei Ngan, and Kayee Hanaoka. *FRVT Part 1: Verification—Ongoing.* NIST Interagency/Internal Reports (NISTIR) 8439. National Institute of Standards and Technology (NIST), 2022. DOI: 10.6028/NIST.IR.8439.

[Led23] Ledger Donjon. *Funds of Every Wallet Created With The Trust Wallet Browser Extension Could Have Been Stolen.* Ledger blog, accessed 20 August 2025. Apr. 2023. URL: https://www.ledger.com/blog/funds-of-every-wallet-created-with-the-trust-wallet-browser-extension-could-have-been-stolen.

[Don+24] Xingbo Dong et al. “WiFaKey: Generating Cryptographic Keys from Face in the Wild”. In: *IEEE Transactions on Instrumentation and Measurement* (2024).

[KZF24] Oleksandr Kuznetsov, Dmytro Zakharov, and Emanuele Frontoni. “Deep learning-based biometric cryptographic key generation with post-quantum security”. In: *Multimedia Tools and Applications* 83.19 (2024), pp. 56909–56938. DOI: 10.1007/s11042-023-17714-7. URL: https://doi.org/10.1007/s11042-023-17714-7.

[Lib+24] Edoardo Liberati et al. “X-Lock: A Secure XOR-Based Fuzzy Extractor for Resource Constrained Devices”. In: *Applied Cryptography and Network Security.* Ed. by Christina Pöpper and Lejla Batina. Cham: Springer Nature Switzerland, 2024, pp. 183–210. ISBN: 978-3-031-54770-6.

[MS24] Daniele Micciancio and Mark Schultz-Wu. *Bit Security: optimal adversaries, equivalence results, and a toolbox for computational-statistical security analysis.* Cryptology ePrint Archive, Paper 2024/1506. 2024. URL: https://eprint.iacr.org/2024/1506.

[Sch+24] Stephanie Schuckers et al. *FIDO Biometrics Requirements*. Technical Report. Final Document, May 22 2024, Editors: Stephanie Schuckers, Greg Cannon, Nils Tekampe, Anthony Lam. FIDO Alliance, May 2024. URL: https://fidoalliance.org/specs/biometric/requirements/Biometrics-Requirements-v4.0-fd-20240522.html.

Appendices

A Security Analysis

A.1 X-Lock Construction Security

In this appendix, we discuss the X-Lock $(\delta, \tau, \varepsilon)$ parameters.

(δ, τ)-correctness We first estimate the correctness of the scheme. Suppose that the user generates the new biometric sample $\mathbf{w}'$ and performs the $\mathsf{Rep}(\mathbf{w}', \mathsf{hs})$ procedure. The procedure succeeds (thus returning $\mathbf{w}$) if and only if the user restores every bit of the entropy β_i, thus:

$$\beta_i = \texttt{MAJ}\left\{\widehat{L}_{i,j} \oplus V_{i,j}\right\}_{j\in[\ell]}, \quad \widehat{L}_{i,j} = \bigoplus_{\alpha\in\mathcal{I}_{i,j}} w'_\alpha$$

When does this happen? This happens if and only if *for each* $i \in [\mu]$ and *for majority* of $j \in [\ell]$, we have $L_{i,j} = \widehat{L}_{i,j}$. We first answer the question: what is the probability of $L_{i,j} = \widehat{L}_{i,j}$ for each i, j? The lockers are the same iff the number of errors $k\omega_H(\{w'_\alpha - w_\alpha\}_{\alpha\in\mathcal{I}_{i,j}})$ is even. To estimate this, we apply the law of total probability:

$$\Pr[\widehat{L}_{i,j} = L_{i,j}] = \sum_{m=0}^{n} \Pr\left[k\omega_H(\{w'_\alpha - w_\alpha\}_{\alpha\in\mathcal{I}_{i,j}}) \text{ is even} \mid \Delta(\mathbf{w}, \mathbf{w}') = \frac{m}{n}\right] q^f\left(\frac{m}{n}\right)$$

The $\omega_H(\{w'_\alpha - w_\alpha\}_{\alpha\in\mathcal{I}_{i,j}})$ is distributed as $\mathrm{Bin}\,(k, m/n)$. To see why, consider the following question: given that we sample k random positions in the binary string (corresponding to the size of $\mathcal{I}_{i,j}$), what is the probability of getting some specific number of errors? Since we have n bits with k error bits, we expect the number of errors to be distributed according to $\mathrm{Bin}(k, m/n)$. That said, the probability of an even number of errors is:

$$\Pr\left[k\omega_H(\{w'_\alpha - w_\alpha\}_{\alpha\in\mathcal{I}_{i,j}}) \text{ is even} \mid \Delta(\mathbf{w}, \mathbf{w}') = \varepsilon\right] = \frac{1}{2} + \frac{1}{2}(1 - 2\varepsilon)^k$$

Substituting the formula into the equation, we have the following:

$$\Pr[\widehat{L}_{i,j} = L_{i,j}] = \sum_{m=0}^{n}\left[\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2m}{n}\right)^k\right] q^f\left(\frac{m}{n}\right)$$

Denote the above probability of guessing the single locker bit by $\theta_L := \Pr[\widehat{L}_{i,j} = L_{i,j}]$. The number of correctly unlocked bits is distributed binomially as $\mathrm{Bin}(\ell, \theta_L)$. The probability of unlocking β_i incorrectly is therefore the probability that this number does not exceed $\approx \ell/2$:

$$\Pr[\beta_i \neq \widehat{\beta}_i] = \sum_{i=0}^{\ell/2}\binom{\ell}{i}\theta_L^i(1 - \theta_L)^{\ell-i}$$

Denote this probability by θ_β. Thus, the correctness value is given by $\delta = 1 - (1 - \theta_\beta)^\mu$.

Remark. *To summarize all the computations, we sequentially estimated:*

- *Probability of unlocking a single locker bit:* $\theta_L := \sum_{m=0}^{n} \left[\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2m}{n}\right)^k\right] q^f\left(\frac{m}{n}\right)$.
- *Probability of not unlocking a single entropy bit:* $\theta_\beta := \sum_{i=0}^{\ell/2} \binom{\ell}{i} \theta_L^i (1-\theta_L)^{\ell-i}$.
- ***Correctness:*** $\delta = 1 - (1-\theta_\beta)^\mu$.

All values are computed numerically based on the provided set of parameters (n, k, ℓ, μ).

To estimate τ, we notice that the aforementioned calculations are the same, except that instead of $q^f(\varepsilon)$, we must use $q^{f,\mathcal{A}}(\varepsilon)$.

Remark. *To summarize all the computations for* τ*:*

- *Probability of unlocking a single locker bit:* $\widetilde{\theta}_L := \sum_{m=0}^{n} \left[\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2m}{n}\right)^k\right] q^{f,\mathcal{A}}\left(\frac{m}{n}\right)$.
- *Probability of not unlocking a single entropy bit:* $\widetilde{\theta}_\beta := \sum_{i=0}^{\ell/2} \binom{\ell}{i} \widetilde{\theta}_L^i (1-\widetilde{\theta}_L)^{\ell-i}$.
- ***Correctness:*** $\tau = (1-\widetilde{\theta}_\beta)^\mu$.

Remark. Note that since $\mathbb{E}_{\varepsilon\sim q^{f,\mathcal{A}}}[\varepsilon] \approx \frac{1}{2}$ and variance of the distribution is low, we can estimate $\widetilde{\theta}_L = \mathbb{E}_{\varepsilon\sim q^{f,\mathcal{A}}}[\frac{1}{2} + \frac{1}{2}(1-2\varepsilon)^k]$ simply as $\frac{1}{2} + \frac{1}{2}(1 - 2\mathbb{E}_{\varepsilon\sim q^{f,\mathcal{A}}}[\varepsilon])^k \approx \frac{1}{2}$. Thus, $\widetilde{\theta}_\beta \approx \frac{1}{2}$ and $\tau \approx 2^{-\mu}$, corresponding to μ-bit level of FAR.

ε-security Let us describe the adversary's $\mathcal{A}$ goal. Given $\mathsf{hs} = (I, V)$ and $\mathsf{sk} = \boldsymbol{\beta} \in \{0,1\}^\nu$, they need to deduce whether $\boldsymbol{\beta}$ was sampled randomly or by computing $\mathcal{H}(\boldsymbol{\beta})$ for some randomly generated blinded entropy $\boldsymbol{\beta}$. Suppose we use Threat Model #1, where the adversary $\mathcal{A}_1$ does not use the model f or information about the underlying distribution $\mathcal{W}_U$. Since $\mathcal{H}$ is modeled as a random oracle, the only adversary's hope is to restore $\boldsymbol{\beta}$ by means of helper data (I, V). Notice that the adversary encounters the system of $\mu\ell$ equations $L_{i,j} \oplus \beta_i = V_{i,j}$ where $L_{i,j}$ and β_i are unknown to the adversary. This way, there are $\mu(\ell+1)$ unknowns in this equation, so it cannot be solved uniquely. The least work the attacker can do is therefore brute-force all possible values of $\boldsymbol{\beta}$, which makes $\varepsilon(\mathcal{A}_1) = 2^{-\mu}$. Another approach that the adversary can try is to solve the system of $\mu\ell$ equations $L_{i,j} = \bigoplus_{\alpha\in\mathcal{I}_{i,j}} w_\alpha$ with respect to $L_{i,j}$, but number of variables is again $n + \mu\ell$, which is greater than $\mu\ell$. Thus, the number of solutions is 2^n (given all equations are independent), which is infeasible to iterate. Thus, we conclude $\varepsilon(\mathcal{A}_1) = 2^{-\mu}$.

Threat Model #2. Naive Attack. In the Threat Model #2, the adversary's $\mathcal{A}_2$ only possibility to distinguish sk from a random string is to recover $\boldsymbol{\beta}$. However, by accessing the oracle $\mathbf{w}' \leftarrow \mathcal{O}^f(\cdot)$, the adversary can try run $\mathsf{Rep}(\mathbf{w}', \mathsf{hs})$ and hope that the obtained entropy would be exactly $\boldsymbol{\beta}$. Then,

$$\varepsilon(\mathcal{A}_2) \geq \Pr\left[\mathsf{Rep}(\mathbf{w}', \mathsf{hs}) = \boldsymbol{\beta} \;\middle|\; \begin{array}{c} \mathbf{w} \leftarrow\!\!\$\; \mathcal{W}_U \\ (\mathsf{hs}, \boldsymbol{\beta}) \leftarrow \mathsf{Gen}(\mathbf{w}) \\ \mathbf{w}' \leftarrow \mathcal{O}^f(\cdot) \end{array}\right]$$

Note that in such a case, we have the same estimate for ε as when computing τ, so $\varepsilon(\mathcal{A}_2) \geq 2^{-\mu}$.

Cascade Attack. Unfortunately, the adversary $\mathcal{A}_2$ can do better than $2^{-\mu}$. One can potentially know which bit is correct by information from the $\mathsf{Rep}(\cdot)$ operation. Here is how. Suppose that the adversary runs $\mathsf{Rep}(\mathsf{hs}, \mathbf{w}')$ using generated $\mathbf{w}' \leftarrow \mathcal{O}^f(\cdot)$. During the computation of each $\widehat{\beta}_i$, the adversary knows the proportion of ones and zeros of μ binary strings $\{\widehat{L}_{i,j} \oplus V_{i,j}\}_{j\in[n]}$. Turns out that with high certainty, if there is a majority bit in some of such strings, the recovered value is likely correct. The adversary stores bits that are more likely to be correct and their position in β. The amount of restored data depends on the oracle function $\mathcal{O}^f(\cdot)$ and the number of conducted attempts q.

The adversary has a probability near 50% to unlock the correct β_i using the feature vector from a different source, including the fact that the adversary could distinguish between incorrect and correct output resulting from $\mathtt{MAJ}(\cdot)$. This means that the adversary needs a small number of attempts to correctly restore the bit. The knowledge of β_i could make it possible to further compromise other bits as well, such as feature vector bits. Consider the following case: The adversary retrieves the correct bit β_{i_0} and gets the system of equations in the form:

$$R_{i_0} : \bigoplus_{\alpha\in\mathcal{I}_{i_0,j}} \mathbf{w}_\alpha = \beta_{i_0} \oplus V_{i_0,j}, \forall j \in [\ell]$$

We want to keep the system underestimated, which means $\ell < n$, and to keep ℓ as small as possible, because the number of solutions will be $2^{n-\ell}$. To have more security, we can set a stricter constraint to choose only once every index from $\mathbf{w}$ in the vault creation step, which means that $\ell \cdot k < n$. When $\ell > n$, we have the overestimated system, which will eventually give the adversary the unique solution for the indices used in the system using the Gaussian elimination method.

If the system is underestimated, the adversary chooses some solution and uses it in $\mathsf{Rep}(\cdot)$. With some certainty, the adversary will know if the restored bit of β is correct. After getting the correct β_i, the adversary could continue to find the solutions to the system or brute-force the unknown bits from the secret value. In case of trying to pick another solution to the system, the adversary could get new unknown β_i until reduced to a smaller number of unknown bits for a brute-force attack or fully restored.

For the case of an overestimated system of equations, the adversary will get the solution if they find n linearly independent equations from the whole R_{i_0}. We guarantee that for ℓ equations there is a unique solution for $\mathbf{w}_i.\forall i \in [n]$. Also, with higher probability, all $\mathbf{w}_i$ will be used in the R_{i_0}, because the system is overestimated. Using Gaussian elimination to find linearly independent equations and solving all of them to retrieve the bits of the feature vector. After this, there are two cases. If all elements of $\mathbf{w}$ were found, then the most effective way of finding β is to use the newly restored feature vector $\mathbf{w}$ in the $\mathsf{Rep}(\cdot)$ procedure. If the found solution for $\mathbf{w}_i, i \in I_{eq} \subset [n]$, then find the equations from the vault, to solve for bits of β. After obtaining all possible β_i, use them to solve for the unknown $\mathbf{w}_i$. There is a possibility that some bits of β are still unknown. Using newly found elements of $\mathbf{w}$, solve for the unknown β_i. Do it until there is no further progress or we restore all values of β and $\mathbf{w}$.

For Threat Model #3, we have the scenario where the adversary $\mathcal{A}_3$ has the photo of the person targeted for the attack. The adversary needs to find correct bits of β and do the procedure as described in the analysis of cascade attack in Threat Model #2.

A.2 Brain-Bio Security Analysis

We start the analysis by estimating the (δ, τ)-correctness of this construction. The user reproduces new N features to extract and generate the set of feature vectors set $\{\mathbf{w}'_s\}_{s\in[N]}$ and a deterministic password as a binary vector $\mathbf{p}$. Using $\mathsf{Rep}(\{\mathbf{w}'_s\}_{s\in[N]}, \mathbf{p}, \mathsf{hs})$, the reproduction result is the entropy value β. Every bit of β could be restored by evaluating the following expression:

$$\beta_i = \texttt{MAJ}\left\{\widehat{L}_{i,j} \oplus V_{i,j}\right\}_{j\in[\ell]}, \quad \widehat{L}_{i,j} = \bigoplus_{s\in[N]} \widehat{L}^{(s)}_{i,j} \oplus P_{i,j}$$

where $\widehat{L}^{(s)}_{i,j} = \bigoplus_{\alpha\in\mathcal{I}^{(s)}_{i,j}} w_{s,\alpha}$ and $P_{i,j} := \bigoplus_{\gamma\in\mathcal{J}_{i,j}} p_\gamma$.

From the (δ,τ)-correctness theorem in Section A.1, we obtained that to successfully restore β_i, one must have the majority of $\widehat{L}_{i,j} = L_{i,j}$ for $j \in [\ell]$. In turn, to successfully restore $\widehat{L}_{i,j}$, the number of incorrect $\widehat{L}^{(s)}_{i,j}$ must be even. Finally, to successfully restore $\widehat{L}^{(s)}_{i,j}$, one needs to have an even number of $k\cdot\omega_H(\{w'_{s,\alpha} - w_{s,\alpha}\}_{\alpha\in\mathcal{I}^{(s)}_{i,j}}$, $s\in[N]$. We assume that the user knows the password $\mathbf{p}$ with 100% certainty. The probability of $\widehat{L}^{(s)}_{i,j} = L^{(s)}_{i,j}$ for legit user is:

$$\begin{aligned}\theta_{L^{(s)}} &:= \Pr[\widehat{L}^{(s)}_{i,j} = L^{(s)}_{i,j}] \\ &= \sum_{m=0}^{n} \Pr\left[k\cdot w_H(\{w'_{s,\alpha} - w_{s,\alpha}\}_{\alpha\in\mathcal{I}^{(s)}_{i,j}, s\in[N]}) \text{ is even} \mid \Delta(\mathbf{w},\mathbf{w}') = \frac{m}{n}\right] q^f_s\left(\frac{m}{n}\right) \\ &= \sum_{m=0}^{n}\left[\frac{1}{2}+\frac{1}{2}\left(1-\frac{2m}{n}\right)^k\right] q^f_s\left(\frac{m}{n}\right)\end{aligned}$$

Here, $q^f_s(\cdot)$ denotes the distance distribution for feature s (with $s\in[N]$).

The probability of getting an even number of errors if the normalized Hamming distance equals some number is the same as in Section A.1. What was changed is the probability of getting the distance values between two feature vectors.

The probability of correctly restoring $\widehat{L}_{i,j}$ is following:

$$\theta_L = \Pr\left[\widehat{L}_{i,j} = L_{i,j}\right] = \frac{1}{2} + \frac{1}{2}\prod_{s\in[N]}(1-2\theta_{L^{(s)}})$$

Using θ_L, we have the following expression for the probability of incorrectly unlocking the β bit:

$$\theta_\beta = \sum_{i=0}^{\ell/2}\binom{\ell}{i}\theta_L^i(1-\theta_L)^{\ell-i}$$

Then, finally $\delta = 1-(1-\theta_\beta)^\mu$.

To estimate τ, we apply the same steps to find the expressions. For the probability of unlocking the correct value of $\widehat{L}^{(s)}_{i,j}$ is:

$$\begin{aligned}\theta_{L^{(s)}} &:= \Pr[\widehat{L}^{(s)}_{i,j} = L^{(s)}_{i,j}] \\ &= \sum_{m=0}^{n} \Pr\left[k\cdot w_H(\{w'_{s,\alpha} - w_{s,\alpha}\}_{\alpha\in\mathcal{I}^{(s)}_{i,j}, s\in[N]}) \text{ is even} \mid \Delta(\mathbf{w},\mathbf{w}') = \frac{m}{n}\right] q^{f,\mathcal{A}}_s\left(\frac{m}{n}\right) \\ &= \sum_{m=0}^{n}\left[\frac{1}{2}+\frac{1}{2}\left(1-\frac{2m}{n}\right)^k\right] q^{f,\mathcal{A}}_s\left(\frac{m}{n}\right)\end{aligned}$$

Here $q_s^{f,\mathcal{A}}(\cdot)$ is the distance distribution for the feature s from the different sources. We assume that the adversary cannot obtain the password $\mathbf{p}$. The probability of correctly obtaining $P_{i,j}$ is 0.5, and the expression of the probability of correctly getting $\widehat{L}_{i,j}$ simplifies to:

$$\widetilde{\theta}_L = \Pr\left[\widehat{L}_{i,j} = L_{i,j}\right] = \frac{1}{2}$$

The evaluation of the probability of incorrectly restoring a bit from β, namely $\widetilde{\theta}_L$, is still the same with bits distributed as $\mathrm{Bin}(\ell, \widetilde{\theta}_L)$:

$$\widetilde{\theta}_\beta := \sum_{i=0}^{\ell/2} \binom{\ell}{i} \left(\frac{1}{2}\right)^i \left(1 - \frac{1}{2}\right)^{\ell-i} = \frac{1}{2^\ell} \sum_{i=0}^{\ell/2} \binom{\ell}{i}$$

The resulting formula for τ is:

$$\tau = (1 - \widetilde{\theta}_\beta)^\mu$$

ε-security As in Section A.1, we estimate the $\varepsilon(\mathcal{A}_1) = 2^{-\mu}$, because the most efficient way of getting the secret values using only the existing vault is a brute-force attack on the key. An attack on the vault is similar to X-Lock construction security analysis, when the adversary tries all possible combinations of β and $L_{i,j}$, where there is no information about the correctness of the solutions to the system of equations in the form $L_{i,j} \oplus \beta_i = V_{i,j}$.

For the Threat Model #2, the presence of password prevents the cascade attacks on the construction with the assumption that the password $\mathbf{p}$ is uniform with 1^λ bits of security. The most efficient way to obtain the secret value when the adversary attempts to reproduce the key is to utilize an oracle, specifically neural networks tailored to each type of object, to determine the feature vector that restores the secret value. From Table 2, we propose the estimation of the security parameter as follows:

$$\varepsilon(\mathcal{A}_2) \geq \prod_{s \in [N]} \mathrm{FAR}^{f,s}$$

A.3 Unforgettable Security Analysis

From Section A.2, Brain-Bio security analysis, the only difference between Unforgettable and Brain-Bio is in the process of obtaining the majority of bits. In Unforgettable $\mathtt{MAJ}(\cdot)$ becomes $\mathtt{MAJ}_\epsilon(\cdot)$. The modification of this part of the construction changes how the probabilities behave with added dependence on the new parameter ϵ. The only change will be in (δ, τ)-correctness and other estimation for ε stays the same.

We are interested in finding the expression for $\theta_\beta(\epsilon)$. The $\mathtt{MAJ}_\epsilon(\cdot)$ Outputs the recovered bit, when $|e_i - \frac{1}{2}| \geq \epsilon$, otherwise $\perp$. To restore the correct bit of β, we need $e_i \in [\frac{1}{2} + \epsilon; 1]$, because we need the majority of correct bits and to be bigger than the threshold. The probability of successfully restoring the correct bit is:

$$\theta_\beta = \sum_{i=(1/2+\epsilon)\ell}^{\ell} \binom{\ell}{i} \theta_L^i (1 - \theta_L)^{\ell-i}$$

Then the correctness value is $\delta = 1 - \theta_\beta^\mu$. Below are outlined all the formulas that have been derived:

- *Probability of unlocking a single locker bit of s feature* $\widehat{L}_{i,j}^{(s)}$: $\theta_{L^{(s)}} := \sum_{m=0}^{n} \left[\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2m}{n}\right)^k\right] q_s^f\left(\frac{m}{n}\right)$.

- *Probability of unlocking a single locker bit* $\widehat{L}_{i,j}$*:* $\theta_L = \frac{1}{2} + \frac{1}{2}\prod_{s\in[N]}\left(1 - 2\theta_{L^{(s)}}\right)$
- *Probability of unlocking a single entropy bit:* $\theta_\beta := \sum_{i=(1/2+\epsilon)\ell}^{\ell} \binom{\ell}{i} \theta_L^i (1-\theta_L)^{\ell-i}$.
- **Correctness:** $\delta = 1 - \theta_\beta^\mu$.

For the adversary case, the calculations are the same. The resulting expressions are:

- *Probability of unlocking a single locker bit of s feature* $\widehat{L}_{i,j}^{(s)}$*:* $\widetilde{\theta}_{L^{(s)}} := \sum_{m=0}^{n}\left[\frac{1}{2} + \frac{1}{2}\left(1 - \frac{2m}{n}\right)^k\right] q_s^{f,\mathcal{A}}\left(\frac{m}{n}\right)$.
- *Probability of unlocking a single locker bit* $\widehat{L}_{i,j}$*:* $\widetilde{\theta}_L = \frac{1}{2}$
- *Probability of unlocking a single entropy bit:* $\widetilde{\theta}_\beta := \frac{1}{2^\ell}\sum_{i=(1/2+\epsilon)\ell}^{\ell}\binom{\ell}{i}$.
- **Correctness:** $\tau = \widetilde{\theta}_\beta^\mu$.

All values are computed numerically based on the provided set of parameters (n, k, ℓ, μ).